Project No. 11-3104

Development and Validation of a Lifecycle-based Prognostics Architecture with Test Bed Validation

Reactor Concepts

Dr. J. Wesley Hines
University of Tennessee-Knoxville

In collaboration with:
Pacific Northwest National Laboratory

Richard Reister, Federal POC
Richard Wood, Technical POC

NEUP Nuclear Energy University Programs
U.S. Department of Energy

Development and Validation of a Lifecycle-based Prognostics Architecture with Test Bed Validation

2014 Final Report

Proposal ID No.: 11-1869

Work Scope Id No.: LWRS-3 – Instrumentation and Control

November 2014

J. Wesley Hines
(Principal Investigator)

Belle Upadhyaya
(Co-Principal Investigator)

Michael Sharp
(Assistant Research Professor)

Pradeep Ramuhalli
(Pacific Northwest National Lab)

Brien Jeffries, Alan Nam, Eric Strong, Matthew Tong, Zachary Welz
(Graduate Research Assistants)

Federico Barbieri
(Visiting Scholar)

Seth Langford, Gregory Meinweiser, Matthew Weeks
(Undergraduate Research Assistants)

The University of Tennessee
Nuclear Engineering Department
315 Pasqua Engineering Building
Knoxville, TN 37996-2300
E-mail: jhines2@utk.edu

Executive Summary

Lifecycle Prognostic Algorithm Development and Application to Test Beds

Introduction:

On-line monitoring and tracking of nuclear plant system and component degradation is being investigated as a method for improving the safety, reliability, and maintainability of aging nuclear power plants. Accurate prediction of the current degradation state of system components and structures is important for accurate estimates of their remaining useful life (RUL). The correct quantification and propagation of both the measurement uncertainty and model uncertainty is necessary for quantifying the uncertainty of the RUL prediction. This research project developed and validated methods to perform RUL estimation throughout the lifecycle of plant components.

Prognostic methods should seamlessly operate from beginning of component life (BOL) to end of component life (EOL). We term this "Lifecycle Prognostics." When a component is put into use, the only information available may be past failure times of similar components used in similar conditions, and the predicted failure distribution can be estimated with reliability methods such as Weibull Analysis (Type I Prognostics). As the component operates, it begins to degrade and consume its available life. This life consumption may be a function of system stresses, and the failure distribution should be updated to account for the system operational stress levels (Type II Prognostics). When degradation becomes apparent, this information can be used to again improve the RUL estimate (Type III Prognostics). This research focused on developing prognostics algorithms for the three types of prognostics, developing uncertainty quantification methods for each of the algorithms, and, most importantly, developing a framework using Bayesian methods to transition between prognostic model types and update failure distribution estimates as new information becomes available. The developed methods were then validated on a range of accelerated degradation test beds.

The ultimate goal of prognostics is to provide an accurate assessment for RUL predictions, with as little uncertainty as possible. From a reliability and maintenance standpoint, there would be improved safety by avoiding all failures. Calculated risk would decrease, saving money by

avoiding unnecessary maintenance. One major bottleneck for data-driven prognostics is the availability of run-to-failure degradation data. Without enough degradation data leading to failure, prognostic models can yield RUL distributions with large uncertainty or mathematically unsound predictions. To address these issues a "Lifecycle Prognostics" method was developed to create RUL distributions from Beginning of Life (BOL) to End of Life (EOL). This employs established Type I, II, and III prognostic methods, and Bayesian transitioning between each Type.

Bayesian methods, as opposed to classical frequency statistics, show how an expected value, a priori, changes with new data to form a posterior distribution. For example, when you purchase a component you have a prior belief, or estimation, of how long it will operate before failing. As you operate it, you may collect information related to its condition that will allow you to update your estimated failure time. Bayesian methods are best used when limited data are available. The use of a prior also means that information is conserved when new data are available. The weightings of the prior belief and information contained in the sampled data are dependent on the variance (uncertainty) of the prior, the variance (uncertainty) of the data, and the amount of measured data (number of samples). If the variance of the prior is small compared to the uncertainty of the data, the prior will be weighed more heavily. However, as more data are collected, the data will be weighted more heavily and will eventually swamp out the prior in calculating the posterior distribution of model parameters. Fundamentally Bayesian analysis updates a prior belief with new data to get a posterior belief.

The general approach to applying the Bayesian method to lifecycle prognostics consisted of identifying the prior, which is the RUL estimate and uncertainty from the previous prognostics type, and combining it with observational data related to the newer prognostics type. The resulting lifecycle prognostics algorithm uses all available information throughout the component lifecycle.

Objectives:

The project had four major objectives related to the development and validation of lifecycle prognostic methods and algorithms. Each of these objectives were completed during the project.

Uncertainty Estimation Quantification:

The Type I, II, and III prognostic models were enhanced to take into consideration all the major sources of uncertainty (initial degradation level, uncertainty inherent in the historical failure time

distribution, measurement uncertainty, and failure threshold uncertainty) and produce a failure distribution output rather than a RUL point estimate. The resulting RUL predictions are failure distributions which inherently consider the combined uncertainty sources.

Lifecycle Prognostic Algorithm Development:

The RUL probability distributions produced by the prognostic models in each stage of the component lifecycle are used as Bayesian prior for the next stage of the lifecycle. The Type I model output is used as a prior for the Type II model, and its output is used as the prior for the Type III model. In cases where stress information (Type II) is not available or useful, Type I outputs are directly used as Type III model priors. These Bayesian transitioning methods were developed and utilized to transition between the three prognostic categories to form a single prognostic system to estimate RUL over the life of the process or component. This is termed Lifecycle Prognostics.

Method Integration and Toolbox Development:

The uncertainty analysis quantification techniques and Bayesian transitioning algorithms were integrated into a MATLAB based Process and Equipment Prognostics toolbox which was merged and integrated into the Process and Equipment Monitoring toolbox resulting in a new toolbox which performs process and equipment monitoring and prognostics (PEMP). This PEMP toolbox was developed with graphical user interfaces and documented with a user manual and tutorial.

Validation of Methods using Accelerated Degradation Test Beds:

Five accelerated degradation component test beds were developed and used to collect data used to validate the Lifecycle Prognostics algorithms. The first three were constructed on-site at the University of Tennessee:

1. A heat exchanger test bed was developed to track degradation due to fouling. The temperatures, flow rates, and pressure were monitored to accurately track the heat transfer as the exchanger degrades.
2. A motor test bed was developed to collect data from 5-HP general-purpose, 3-phase, 3600 RPM induction motors. The motors were subjected to cyclic thermal aging processes designed to induce accelerated insulation breakdown and corrosion within the motors. These tests were run for close to 18 months with temperatures increased during the last few months. Several parameters such as vibration, motor current, motor voltage, and acoustic data were collected. The insulation of the incoming wires was found to be

the weak point resulting in sudden, random failures, and Type III prognostics was not successful. Additional testing using motor overload to incite insulation degradation and winding-to-winding short is continuing.

3. The third test setup consisted of neoprene pump impellers, which were subjected to high levels of heat stress. During testing, the vibration, differential pressure, and current of the pump were monitored to track degradation of the impellers at different operating conditions.

At Pacific Northwest National Laboratory (PNNL), a passive component material test bed was constructed to generate degradation data for fatigue of metal components. These data were shared with the University of Tennessee, and prognostic algorithms proved successful. A fifth test bed for bearing failure by Analysis and Measurement Services Corporation (AMS) was developed and data provided to UT was also used to validate the prognostic algorithms. All five test beds resulted in run-to-failure degradation data that were used to validate the lifecycle prognostic modeling algorithms. This project successfully developed and validated a holistic Lifecycle Prognostics method, in which Bayesian transitions were applied.

Additional Accomplishments:

The following is a summary of additional accomplishments made during this research:

- The project PI and the Co-PI were engaged in three synergistic research projects. One of these was a DOE STTR Phase-1 project with the industrial partner, AMS, with an emphasis on Small Modular Reactor (SMR) fault monitoring. Phase-2 of this project is on-going with the University of Tennessee as a collaborating institution. The second involved collaboration with Korea Atomic Energy Research Institute (KAERI) on a DOE I-NERI project and was related to monitoring safety critical functions during beyond design basis accidents in light water reactors. The latter also addresses station blackout, remote sensing, self-powered detectors, energy harvesting, and wireless communication. The third collaborative project was titled *Integral Inherently Safe Light Water Reactor (I2S-LWR)*, with Georgia Institute of Technology as the lead for this NEUP-IRP. These indicate the value of this NEUP project in enhancing the safety and operational reliability of both existing and future nuclear power plants.
- Several funded students completed their MS and PhD degrees:
    - Eric Strong (MS Reliability and Maintainability Engineering, 2013)

- Tutpol Ardsomang (M.S. Reliability and Maintainability Engineering, 2013, funding by home country)
- Alan Nam (M.S. Nuclear Engineering, 2013)
- Matthew Tong (M.S. Nuclear Engineering, 2013)
- Eric Strong (Ph.D. Nuclear Engineering, 2014) Dissertation title: "Development of a Method for Incorporating Fault Codes in Prognostic Analysis"
- Alan Nam (Ph.D. Nuclear Engineering, expected May 2015) Dissertation title: "Bayesian Optimization for Fusion of Multiple Prognostic Methodologies"

- Journal articles published or in press:
  1. Nam, A, M. Sharp, B.R. Upadhyaya, and J.W. Hines, "Lifecycle Prognostic Algorithm Development and Application to Test Beds", *Chemical Engineering Transactions*, Vol. 33, 2013
  2. Sharp, M., Coble, J., Nam, A., Hines, J., and, Upadhyaya, B. "Lifecycle Prognostics: Transitioning Between Information Types", Invited by the Journal of Risk and Reliability, 2014, In Press.
- Journal articles invited or submitted:
  1. Sharp, M., and J.W. Hines, "Improved Metrics for Evaluation of Prognostic Model Performance", Submitted to the Prognostics and Health Management Journal, 2013.
  2. Welz, Z., M. Sharp and J.W. Hines, "Prognostics for Nuclear Power Plant Life Extension", Invited by the International Journal of Prognostics and Health Management (IJPHM) 2014.
  3. Barbieri, F., J.W. Hines, M. Sharp, and M. Venturin, "Sensor-Based Degradation Prediction and Prognostics for Remaining Useful Life Estimation – Validation on Experimental Data of Electric Motors", Submitted to the International Journal of Prognostics and Health Management (IJPHM) 2014 special issue on Nuclear Energy PHM, October, 2014.
- External Reports:
  1. Coble, J.B., P. Ramuhalli, L.J. Bond, J.W. Hines, B.R. Upadhyaya, "Prognostics and Health Management in Nuclear Power Plants: A Review of Technologies and Applications", Pacific Northwest National Laboratory Report, PNNL-21515, July 2012.
  2. "Advanced Surveillance, Diagnostic and Prognostic Techniques in Monitoring Structures, Systems and Components in Nuclear Power Plants", IAEA Technical Report No. NP-T-3.14, 2013.

- Conference publications:
  1. Tong, M., J.W. Hines, B.R. Upadhyaya, and M. Sharp, "Application of POF Distributions and RUL Estimates", 8th NPIC & HMIT, July 2012, San Diego, California.
  2. Nam, A., J.W. Hines and B.R. Upadhyaya, "Bayesian Methods for Successive Transitioning Between Prognostic Types", 8th NPIC & HMIT, July 2012, San Diego, California.
  3. Hines, J.W., M. Tong, A. Nam, M. Sharp, and B. Upadhyaya, "Lifecycle Prognostics for Asset Management", Maintenance and Reliability Conference, MARCON, Knoxville, TN, 2013.
  4. Nam, A., J.W. Hines and M. Sharp, "Development of Bayesian Transitions for Data-Driven Prognostics", PHM Conference, New Orleans October 14-17, 2013.
  5. Hines, J.W. and B.R. Upadhyaya, "Life-Cycle Prognostics for the LWRS Program", *Light Water Reactor Sustainability Newsletter,* Idaho National Laboratory, Issue 10, January 2013, pp. 2-5.
  6. Nam, A., M. Sharp, B.R. Upadhyaya, and J.W. Hines, "Lifecycle Prognostic Algorithm Development and Application to Test Beds", 2013 Prognostics and System Health Management Conference PHM-2013, Milan, 2013.
  7. Ardsomang, T. J.W. Hines, and B.R. Upadhyaya, "Heat Exchanger Fouling and Estimation of Remaining Useful Life", Annual Conference of the PHM Society, New Orleans, LA, October 2013.
  8. Nam, A., M. Sharp, J.W. Hines, B.R. Upadhyaya, "Lifecycle Prognostic Algorithm Development Using Bayesian Statistics", Transactions of the 2013 Winter ANS meeting, Nov, 2013.
  9. Welz, Z., Nam, A, Sharp, M., J. Hines, and B. Upadhyaya, "Prognostics for Light Water Reactor Sustainability: Empirical Methods for Heat Exchanger Prognostic Lifetime Predictions", 2nd European Conference of the Prognostics and Health Management Society (PHME'14), Nantes, France, July 8-10, 2014. (Best Paper Award)
  10. Jeffries, B., Hines, J.W., A. Nam, M. Sharp, and B.R. Upadhyaya, "Lifecycle Prognostic Algorithm Development and Initial Application Results", ISOFIC/ISSNP 2014: International Symposium on Future I&C for Nuclear Power Plants/International Symposium on Symbiotic Nuclear Power Systems, Jeju Island. Korea, August 2014.
  11. Coble, J.B., P. Ramuhalli, L.J. Bond, J.W. Hines, B.R. Upadhyaya, "Summary of Nuclear Power Plants Prognostics and Health Management Report: PNNL-21515", ISOFIC/ISSNP 2014: International Symposium on Future I&C for Nuclear Power

Plants/International Symposium on Symbiotic Nuclear Power Systems, Jeju Island. Korea, August 2014.

- Articles in the Popular Press:
  1. Hines, J.W. and B.R. Upadhyaya, "Life-Cycle Prognostics for the LWRS Program", *Light Water Reactor Sustainability Newsletter,* Idaho National Laboratory, Issue 10, January 2013, pp. 2-5.
- Conference publications accepted for publication:
  1. Welz, Z., A. Nam, M.J. Sharp, J.W. Hines and B. R. Upadhyaya, "Optimization Techniques for Prognostic Parameter Generation", NPIC&HMIT2015, Charlotte, NC, 2015.
  2. Barbieri, F., J.W. Hines, M. Sharp, and M. Venturin, "Sensor-Based Electrical Motor Degradation Prediction and Prognostics for Remaining Useful Life Estimation", NPIC&HMIT2015, Charlotte, NC, 2015.
  3. Nam, A., B.A. Jeffries, J.W. Hines, B.R. Upadhyaya, "Prognostic Methodologies for Pump Impeller Lifecycle Degradation NPIC&HMIT2015, Charlotte, NC, 2015.

Table of Contents

Executive Summary 2

1. Introduction 16

1.1 Background and Project Objectives 16

1.2 Research Tasks 19

1.3 Report Organization 21

2. Task 1: Uncertainty Estimation Techniques 22

2.1 Literature Survey 22

2.1.1 Data Simulation 22

2.1.2 General Path Model Estimations and Associated Uncertainty 25

2.1.3 Monte Carlo Simulations 27

2.2 Uncertainty 27

2.2.1 Type I Theory 27

2.2.2 Application 28

2.2.3 Type II 31

2.2.4 Markov Chain Theory 31

2.2.5 Markov Chain Application 32

2.3 Type III 36

2.3.1 Type III GPM Model Types 36

2.3.2 Type III GPM Model Methodologies 37

2.3.3 Model Uncertainty 38

2.3.4 Model Transformations 43

2.4 Non-Linear Uncertainty 47

3. Task 2: Lifecycle Prognostics 51

3.1 Bayesian and Classical Statistics 52

3.1.1 Gaussian Conjugate Distributions 53
3.1.2 Weibull Sampling Distribution 54
3.1.3 OLS Regression 54
3.1.4 Linear Regression with Bayesian Priors 54
3.1.5 Local Linear Regression and Locally Weighted Regression 55
3.2 Transitions between Prognostics Types 56
3.2.1 Type I to Type II Transition 56
3.2.2 Type I/II to Type III Transition 57
3.2.3 Alternatives to GPM via Local Models 57
3.2.4 Alternative Methods Conclusions 59
4. Task 3: Performance Metrics 61
4.1 Performance Metric Demonstration 61
4.1.1 Mean Absolute Percent Error 62
4.1.2 α-λ Performance 62
4.1.3 Prognostic Horizon 63
4.1.4 Relative Accuracy (RA) and Cumulative RA (CRA) 64
4.1.5 Convergence 66
4.1.6 Uncertainty Spread 67
4.1.7 Remarks on Metrics 68
4.2 Performance Metrics Methodology 68
4.2.1 Performance Metric Hierarchy 69
4.2.2 Performance Metric Evaluation 70
4.2.3 Remarks 74
5. Task 4: Integrate PEP with Bayesian Transitioning 75
5.1 Modifications to PEP 75
5.2 Bayes Transitions Functionality 75
5.2.1 Type I to Type II in PEP 76

5.2.2 Transitioning to GPM in PEP 77
6. Task 5: Integrate the PEP and PEM Toolboxes 79
6.1 Introduction 79
6.2 The Process Equipment Monitoring and Prognostics Suite 80
6.2.1 Cox Proportional Hazards and General Path Model Transitioning Using Bayes 81
6.2.2 GPM 81
6.2.3 OLS with Bayes 82
7. Task 6: Performance Metrics Development 84
7.1 Mean Absolute Error 84
7.2 Weighted Error Bias 85
7.3 Percent Error Value Binning 86
7.4 Weighted Prediction Spread 87
7.5 Confidence Interval Coverage 88
7.6 Confidence Convergence Horizon 89
7.7 Total Score Metric 90
7.8 Prediction Metric Example Cases Studies 91
7.9 Summary and Conclusions 95
8. Task 7 and Task 8: Experimental Setups 97
8.1 Electric Motor Aging Experiment 97
8.1.1 Experimental Methodology 98
8.1.2 Initial Fault Mode Forensics 102
8.1.3 Motor Aging Experiment Test Procedure Modifications 103
8.1.4 Phase II Motor Selection 104
8.1.5 Motor Aging Experiment Results 105
8.1.6 Motor Aging Mechanical Overload Testing 106
8.2 Large Neoprene Impeller Degradation Experiment 107
8.2.1 Literature Review 108

8.2.2 Physical Setup....113
8.2.3 Thermal Aging Procedures:....114
8.3 Heat Exchanger Fouling Experiment....115
8.3.1 Experimental Setup....115
8.3.2 Experiment Modifications for Improved Degradation Cycles....117
8.3.3 Status of Experiment/Operating Conditions....121
8.3.4 Data Pre-processing....121
8.3.5 Experimental Results....122
8.4 Passive Material Test Bed....123
8.4.1 Nondestructive Evaluation Measurements....124
8.4.2 Bench-Scale Test Beds....125
8.5 Offsite Bearing Testing Facilities....128
8.5.1 Experiment Setup....129
8.5.2 Data Analysis....130
8.6 TASK 8: Degradation and Failure Data Collection....135
9. Task 9: Use Algorithms to Develop Lifecycle Models....142
9.1 Analysis of Accelerated Motor Aging Data....142
9.1.1 Experiment & Data Description....142
9.1.2 Feature Extraction: Time Domain....143
9.1.3 Feature Extraction: Frequency Domain....150
9.1.4 Modeling and Prognostic Parameter Development....155
9.1.5 Prognostic Model Results....159
9.2 Large Impeller Degradation Experiment....168
9.2.1 Type I Prognostic Model Results....169
9.2.2 Type II Prognostic Model Results....170
9.2.3 Type III Prognostic Model Results....172
9.3 Analysis of Heat Exchanger Data....178

9.3.1 Signal and Feature Sets....178
9.3.2 Auto-Associative Kernel Regression Model....181
9.3.3 Prognostic Parameter Generation....183
9.3.4 General Path Model and Bayesian Updating....184
9.3.5 Bayes Method Implementation....186
9.3.6 Results and Discussion....188
9.3.7 Finishing Efforts....191
9.4 Barkhausen Noise Measurements....191
9.4.1 Analysis....191
9.5 Non-Linear Ultrasonic Measurement System....194
9.5.1 Time Series Investigation Results....195
9.5.2 Frequency Analysis....198
9.5.3 Generic Pattern Matching....200
10. Task 10 and Task 11: Validation of Developed Procedures....203
10.1 General Transitioning Procedure Example....203
10.1.1 Type I to Type II....203
10.1.2 Transitioning Example Type I/II to GPM Transitions....205
10.2 Pump Impeller Example Case....209
10.2.1 Type I Prognostic Model Development....209
10.2.2 Type II Prognostic Model Development....210
10.2.3 Type III Prognostic Model Development....213
10.3 Monitoring and Prognostics Tutorial....214
10.4 Summary:....227
11. Concluding Remarks and Summary....228
12. Future Work:....230
13. References....235
14. Appendix 1: Additional Heat Exchanger Figures....237

14.1 Plot of Features 237
14.2 Summary of Signal Indices 237
14.3 Code for LabView Extraction 238
15. Appendix 2: PEMP Suite Users guide 241
15.1 STEP 1 – Import and Separate Data 241
15.2 Step 2 – Create a New Project 242
15.3 Step 3 – Build Signal Monitoring Model 246
15.4 Step 4 – Prognostic Lifetime Prediction Model Creation 251
15.5 Step 5 – Run Query Data 254
16. Appendix 3: PEM and PEP Users Guide 258
16.1 Introduction 258
16.1.1 System Requirements 258
16.1.2 Toolbox Requirements 258
16.1.3 Toolbox Architecture 259
16.1.4 Release Notes 259
16.2 Background 261
16.2.1 Monitoring 264
16.2.2 AAKR 265
16.2.3 Performance Metrics 265
16.2.4 Type I Prognostics 266
16.2.5 Type II Prognostics 268
16.2.6 Markov Chain Models 268
16.2.7 Shock Models 270
16.2.8 Proportional Hazards Models 270
16.2.9 Type III Prognostics 272
16.2.10 Prognostic Parameter 273
16.2.11 General Path Model 273

16.2.12 Particle Filtering Model 278
16.2.13 Selecting a Prognostic Parameter 281
16.2.14 Bayesian Transitions 283
16.2.15 Type I to Type II Transition 284
16.2.16 TypeI/II to Type III Transition 285
16.3 PEM Functions List 288
16.4 Appendix 3 References 442

1. Introduction

1.1 Background and Project Objectives

On-line monitoring of nuclear plant system degradation is quickly becoming a crucial consideration as the licenses of many nuclear power plants are being extended. Accurate measurement of the current degradation of system components and structures is important for correct estimates of their Remaining Useful Life (RUL). The propagation of the uncertainty involved in both the measurements and model construction of these system components is vital in finding the uncertainty of the overall RUL calculation of the system.

Prognostics involves predicting the amount of time or cycles that a system component will continue to meet its design specifications. The amount of time until a system component fails to meet its design specifications is also called the RUL. A useful way to specify the RUL is not merely by a point estimate, but rather by a distribution of the component's probability of failure. During the early life of the component, the probability of failure might be very low, while a component later in life will have a much higher probability of failure. One of the goals of this project is to develop techniques for using uncertainty in the measurement and system models to construct the probability of failure distributions.

On-line, continuous, or periodic equipment health monitoring systems are made up of several modules that solve specific tasks. Data collected from the component or system of interest is used for monitoring purposes. Component monitoring usually consists of comparing current data to expected normal operational data to detect small changes that are related to equipment degradation. The onset of equipment degradation is commonly called a fault. When a fault is detected, a diagnosis module is activated to identify the location and type of fault. Fault identification is important because different faults progress to failure in different and predictable ways and therefore require different prognostic models. Once the fault is identified, a prognostic model is activated which uses information such as current and past environmental and usage conditions, current and past operational sensor data, and historical failure data for like components to predict an item-specific probability of failure distribution.

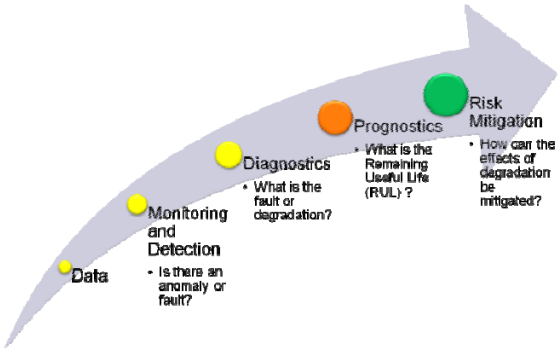


Figure 1.1-1 Prognostics and Health Management System

Prognostics methods can be categorized by their architecture, how they operate, the results format they produce, or through several other means. An approach that may be most instructive is to categorize them by the type of information they use to make RUL estimates, resulting in three prognostic method types. The most common is Type I, which is the topic of most reliability engineering texts. Additionally, Type II and Type III prognostics use operational and condition data to provide more accurate and precise RUL predictions. The three prognostic types are briefly discussed below.

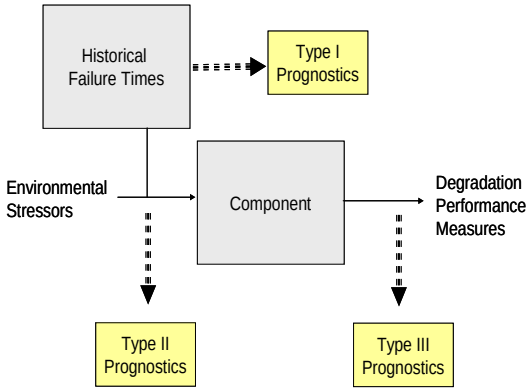


Figure 1.1-2 Prognostic Method Types

Type I: Reliability Data-based Prognostics

These methods consider historical time to failure data, which are used to model the failure distribution for current system times. They estimate the life of an average component under average usage conditions. The most common method is Weibull Analysis and has been well studied for several decades.

Type II: Stress-Based Prognostics

A readily apparent disadvantage of reliability data-based prognostic models (Type I) is that they do not consider the operating condition of the component. However, components operating under harsh conditions would be expected to fail sooner and components operating under mild conditions to last longer than the average failure time. A group of prognostic methods that takes the operating conditions under consideration is aptly named stress-based prognostics. Examples include the proportional hazards model and Markov Chain (MC) models.

Type III: Effects-Based Prognostics

Effects-based prognostic models use degradation measures to form a prognostic prediction. A *degradation measure* is a scalar or vector quantity that numerically reflects the current ability of the system to perform its designated functions properly. It is a quantity that is correlated with the probability of failure. A *degradation path* is a trajectory along which the degradation measure evolves in time towards the critical level corresponding to a failure event. Effects-based prognostics models track the degradation (damage) as a function of usage and predict when the total damage will exceed a predefined threshold that defines failure. There are several mathematical approaches to model cumulative damage such as Markov Chain-based models, general path models, and particle filters.

For prognostics to be useful, the methods should seamlessly operate from beginning of component life (BOL) to end of component life (EOL). We term this "**Lifecycle Prognostics.**" When a component is put into use, the only information available may be past failure times, and the predicted failure distribution can be estimated with reliability methods such as Weibull Analysis (Type I). As the component operates, it begins to consume its available life. This life consumption may be a function of system stresses and the failure distribution should be updated (Type II). When degradation becomes apparent, this information can be used to again improve the failure distribution estimate (Type III). Current research focuses on developing methods for the three types of prognostics. This research project will develop a framework using Bayesian methods to transition between prognostic model types and update failure distribution estimates as new information becomes available.

The four primary objectives of this research included:

- Develop methods to formulate and integrate models from the three prognostics categories into a single prognostic system to estimate RUL over the life of the component.

- Develop techniques to estimate uncertainty and produce a failure distribution output from each of the prognostic model types.

- Integrate the models and methods developed in objective 1 into a toolset to provide a formal method for prognostics development that can be used for prognostics in general, whether it is for active and passive components or systems.

- Validate the methods on a range of test beds. Several test beds that were constructed to validate the Process and Equipment Monitoring (PEM) algorithms were also used to validate the Process and Equipment Prognostics (PEP) algorithms.

Each of these objectives was successfully completed.

1.2 Research Tasks

Listed below are the milestone tasks and a short description of their completion.

> **Task 1.** RUL uncertainty estimation techniques were developed for each model type resulting in POF distributions rather than point estimates. Methods were developed for Type II and III prognostics that incorporate stressor measurement uncertainty prognostic parameter uncertainty, model uncertainty, and model misspecification to produce POF distributions.
>
> **Task 2.** A Bayesian method for transitioning between POF distributions for Type I, II, and III methods was developed. This is termed Lifecycle Prognsotics.
>
> **Task 3**. Performance measures were developed for the prognostics models. Recently, NASA has reported some potential ideas to assess prognostic model performance. This task integrated those performance metrics, along with several developed by the research team, into the prognostic model development framework so that models could be optimized and model algorithm performance could be compared.

The second year tasks consisted of the following:

> **Task 4.** The prognostic algorithms and their Bayesian updating and transitioning algorithms were integrated into a Process and Equipment Prognostics (PEP) Toolbox to provide a formal method for prognostics development that can be used for prognostics in general, whether it is for active or for passive components or systems (electronics, structure, equipment, etc.)

Task 5. The PEP toolbox was integrated with the Process and Equipment Monitoring (PEM) toolbox resulting in a powerful set of tools that could be used to develop models for on-line equipment condition monitoring, anomaly detection, anomaly interpretation, and prognostics for both passive and active components.

Task 6. Metrics and criteria were developed to select appropriate and optimal prognostic models for specific applications and sets of data.

Task 7. Three test beds were specified and constructed for prognostic algorithm validation. These included accelerated degradation test beds for motors, pumps, and a heat exchanger.

The third year tasks were:

Task 8. The three test beds were used to collect environmental stressor, degradation and failure data. These tests took on the order of days for the pumps, weeks for the heat exchanger, and months for the motors.

Task 9. The PEM and PEP Toolboxes were used to develop and optimize monitoring and lifecycle prognostic models for the collected data. The MATLAB-based development suite was used to develop detection and identification modules, and construct prognostic information rich features. The PEP toolbox was used to develop the lifecycle based prognostic models with Type I POF distribution estimation, Type II model development using environmental information, optimal prognostic parameter generation, and Type III prognostic model development. Bayesian methods were used to update estimates and transition between model types.

Task 10. The developed prognostic methods were validated using the collected data sets.

Task 11. Procedures and a user manual for the MATLAB-based toolsets were developed for implementation of the lifecycle prognostic algorithms.

Task 12. Finally, a final report was prepared.

1.3 Report Organization

Each chapter after the introduction enumerates the assigned project tasks, details of work scope and a summary of work completed are provided. The final chapter includes a summary and recommendation for future work. The appendix contains user guides for the MATLAB toolboxes developed for monitoring and prognostics.

2. Task 1: Uncertainty Estimation Techniques

There are primarily four sources of uncertainty that are important to assess during prognostic modeling: initial degradation level, uncertainty inherent in the historical failure time distribution, measurement uncertainty, and failure threshold uncertainty. By understanding and accounting for these sources of uncertainty, researchers will be able to not only create more accurate models, but also provide a better assessment of the predictions and information provided with those models. Both literature surveys and data method investigations into proper quantification and propagation of the sources of uncertainty within various types of model were performed in the completion of this task. The primary focus of this investigation centered on, but was not strictly limited to; online effects based (Type III) prognostic models including the General Path Model (GPM) with and without Bayesian updating. The results of this investigation are provided in this section.

2.1 Literature Survey

At the start of investigation into this task, a literature survey was carried out to form the basis of analysis. The topics included the generation of simulated data, regression analysis applied to feature extraction, Monte Carlo simulations of prognostic models, and other applied prognostic methods.

2.1.1 Data Simulation

A key aspect of this study is the development and implementation of prognostic models that simulate and demonstrate the component degradation through time. According to a paper by Laskey in 1996, there are two types of models that can be used for research purposes.

The first type is an exploratory model, which lays the theoretical groundwork for identifying what needs to be determined and by what method. A good example is the use of simulated data based on algebraic functions, varied by normally distributed parameters. The data that will be used to test the algorithms is idealized, in that the underlying properties are clearly defined and the properties of the algorithm can be researched and understood using the model.

The second type is referred to as a consolidatory model. The consolidatory model is more practical and generally useful for more detailed experiments that may replace an actual system. The key difference between exploratory and consolidatory modeling is that exploratory modeling is designed to suit the test algorithm, while a consolidatory model is designed to simulate a real world system. An example of this is simulating a virtual water pump loop based on first principle

models. The outcome of applying the algorithm will show an expected result if applied theoretically to a real world system.

This project focuses on primarily exploratory modeling, identifying the relationships between various model assumptions and outputs. An example of this is a basic linearly increasing degradation model, which will be defined later in this section.

Equation 2-1 $\ln(x) = \varepsilon + \beta_1 + \beta_2 x$ Equation 2-2 **Equation 2-1**

Equation 2-2 shown below, was used to create the simulated data, which will later be modeled using a General Path Model (GPM). General Path Modeling is a method of data extrapolation through parametric function fitting and linear regression. Further explanation of the use of the GPM is discussed later in this section. It is important to select the correct type of GPM based on the data, such as linear, quadratic, or exponential, because an improperly chosen model not suited to a particular data set can negatively affect the uncertainty values of the model predictions. For this study the simulated data was constructed using the basic linear equation.

$$Y(x) = \varepsilon + \beta_1 + \beta_2 x \qquad \textbf{Equation 2-1}$$

In Equation 2-1, Ɛ is a normally distributed stochastic error, β_1 is a random parameter chosen to represent initial degradation, and β_2 is a normally-distributed random parameter chosen to represent the rate of degradation. The data used in this section has the population parameters normally distributed about β_1= 0.2, β_2=1.2, and Ɛ=0. The linear model shown in the equation is commonly used and can be found detailed in other sources such as Nagi 2009 and Wiesel 2008. Using this equation, a simulated historic population of degradation pathways was created and is shown in Figure 2.1-1.

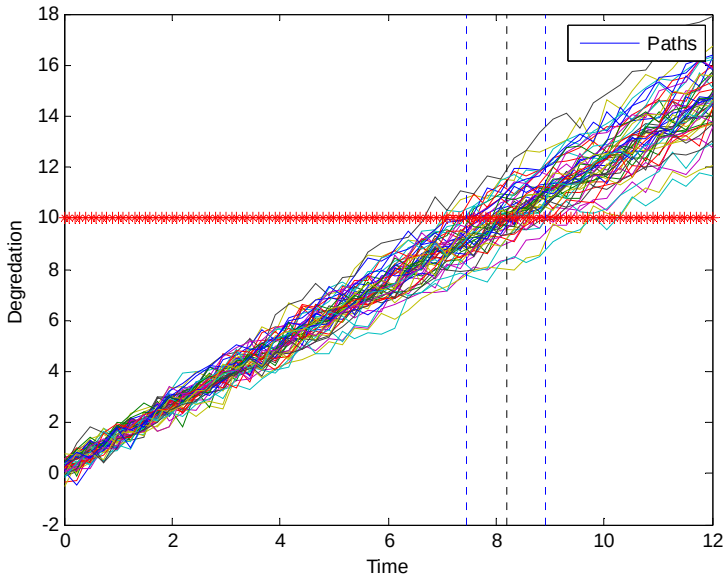


Figure 2.1-1 Linear General Path Model

The fifty historic degradation pathways in Figure 2.1-1 represent the lifecycle of a single simulated component from the total population. The red horizontal line crossing the model paths is the selected critical degradation threshold, beyond which the component is considered to have failed or no longer meets specifications. In this example the threshold is a hard limit for defining failure, but the idea of a soft and failure limit distributions will be explored in later sections.

Although linear models are often found in real world applications, it is important to note that other model types, such as quadratic and exponential may also be useful in particular applications. The functional form of an exponential model developed by Coble (2010) is shown in Equation 2-2.

$$\ln(x) = \varepsilon + \beta_1 + \beta_2 x \qquad \textbf{Equation 2-2}$$

Figure 2.1-2 shows an example of an exponential GPM that was developed with fifty degradation paths. As in the previous example, the red line represents a critical failure threshold, the components are considered to have failed after the degradation paths cross this line.

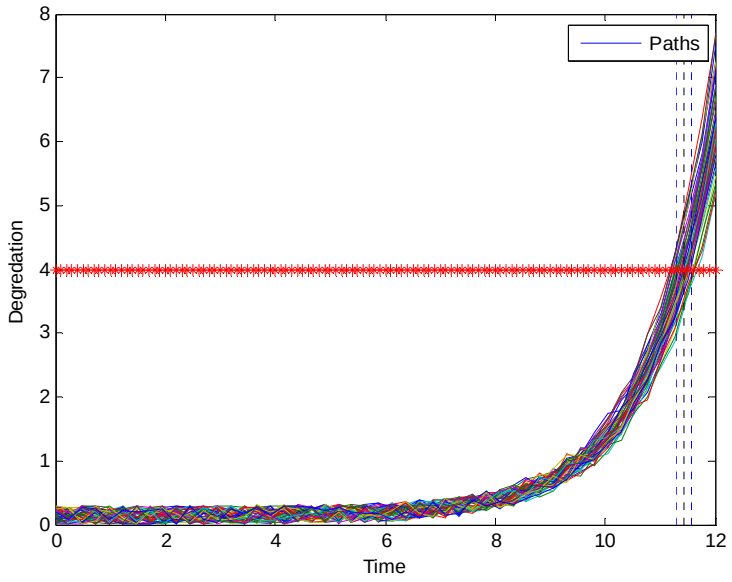


Figure 2.1-2 Exponential General Path Model

2.1.2 General Path Model Estimations and Associated Uncertainty

The GPM approach to prognostics applies a known functional form to a degradation pathway and extrapolates to a critical failure threshold. Important assumptions of the GPM are that the degradation paths can be fitted to a known functional form and that there is a critical degradation threshold. The model can take the form of any dominantly monotonic function across the region of interest. Some examples of this include linear, quadratic, exponential, etc. A critical degradation or failure threshold is also required, the crossing of which indicates failure. This serves as the target that the paths are extrapolated to in order to determine estimated failure times.

RUL estimates are one of the values of interest when using a prognostic algorithm. In order to place an appropriate level of confidence in these estimates, RULs should be accompanied by an estimation of the associated calculated uncertainty. Uncertainty is present in a multitude of sources, each of which should be understood and analyzed in order to find the total uncertainty in any RUL estimation. This research focuses on the sources of uncertainty that are associated with the GPM. Calculating the parameters for the GPM is accomplished using a combination of Bayesian linear regression and Ordinary Least Squares (OLS). These methods and how they affect uncertainty will be discussed in this section.

Bayesian Updating:

Bayesian linear regression is a method that can use prior assumptions or information along with current data to estimate the parameters in a regression model. Bayesian uncertainty analysis is based on Bayes' Theorem, which is presented in **Equation 2-3** [Leon 2008].

$$P(\theta|Y) = \frac{P(\theta)P(Y|\theta)}{P(Y)} \qquad \textbf{Equation 2-3}$$

In this equation θ represents the model parameters and Y is a set of independent measurements. P(θ) is the prior distribution, P(Y|θ) is the likelihood function, and P(Y) is the integral of P(θ)P(Y | θ) over all of θ. A full discussion of the Bayesian linear regression method as applied to RUL estimation can be found in Usynin 2007.

Other forms of Bayesian updating and Bayes theorem exist and can be applicable in particular instances, such as Bayesian Inference [Engel 2000]. Another application uses Bayesian updating with a particle filter example [Saha and Goebel, 2009]. Although these are not discussed in detail here, their usefulness is well documented in other publications.

Ordinary Least Squares:

The OLS method is an error minimization technique commonly used to estimate unknown parameters in a linearly transformable regression model. The regression parameters are calculated using the least squares equation.

$$\beta = (X^T X)^{-1} X^T Y \qquad \textbf{Equation 2-4}$$

Using these parameters with the GPM allows for extrapolation and measurement error estimation. Further information can be found in Usynin 2007.

Population Estimates:

Another method for RUL estimation is direct calculations, otherwise known as population estimates. Population estimates can be essentially represented by mean and variance calculations of RUL times, but there are a variety of ways to illustrate such estimates. A good source on RUL uncertainty can be found in Engel 2000. Engel analyzes four types of probability density function (PDF) applications for RUL. These four types are: a true PDF priori at time zero; use of a modeled PDF to predict a true PDF at time zero; a true posteriori PDF conditioned using

observations taken during use of a component; and creation of a modeled PDF that estimates a true posteriori PDF while component is in use.

2.1.3 Monte Carlo Simulations

Monte Carlo is an estimation method that employs repeated simulation trials to produce a distribution of probable results. Due to the fact that a distribution is inherently produced during this process, Monte Carlo is very useful as a means for estimating uncertainty. Wang 2011 and Jie 2011 give methods for calculating uncertainty using Monte Carlo. The uncertainty calculation applies multiple model iterations and taking the mean and standard deviation of the individual results to find a point estimate and uncertainty.

2.2 Uncertainty

Understanding the uncertainty associated with RUL estimates is an important area of prognostics. For each of the three types of prognostic models, there exist different methods to produce RUL and uncertainty estimates. Uncertainty estimates can be in the form of 95% confidence intervals and using PDFs,, as found in Meeker and Wiley 1998. The following section will detail common estimation techniques for producing RUL and uncertainty estimates for each type of prognostic model.

The three categories of prognostic methodologies are identified as Type I, II, and III and are differentiated by the type of information used to produce the estimate. Type I estimates are the more traditional failure time analyses. They are most applicable when only historical failure time data is available. Type II estimates employ conditions and states to augment predictions. Lastly, Type III uses degradation information in the form of a health indicator of a system or component to obtain RUL estimates.

2.2.1 Type I Theory

Type I analysis is defined as the study of lifetime data, including both known failure times and censored failure times. This method characterizes the average component lifetime and assumes that future components will fail on a similar time scale based of the past failure time distribution. In the following example, the failure times of a set of components will be fitted to a distribution, and the distribution will be used to determine the conditional probability of the component being tested. This has the form of **Equation 2-5** (Coble, 2011) with R being reliability.

$$R(t|t > T) = \frac{R(t)}{R(T)}$$ **Equation 2-5**

The RUL is determined by the Mean Residual Life (MRL) in **Equation 2-6**, where S is survivability and t is time. This function adjusts according to time already spent operating, as seen by the lower limit t of the function.

$$MRL(t) = \frac{1}{S(t)} \int_{t}^{\infty} S(u)du$$ **Equation 2-6**

The Mean Residual Life is calculated at the point where 50% of the components are expected to fail while adjusting for the current time the component has already operated.

2.2.2 Application

To demonstrate the use of Type I conditional reliability, Figure 2.2-1 shows a selection of failure time data, which is the only data needed for Type I analysis. The figure shows the failure times and frequency of 135 units, which had average failure times between 100 and 140 time units.

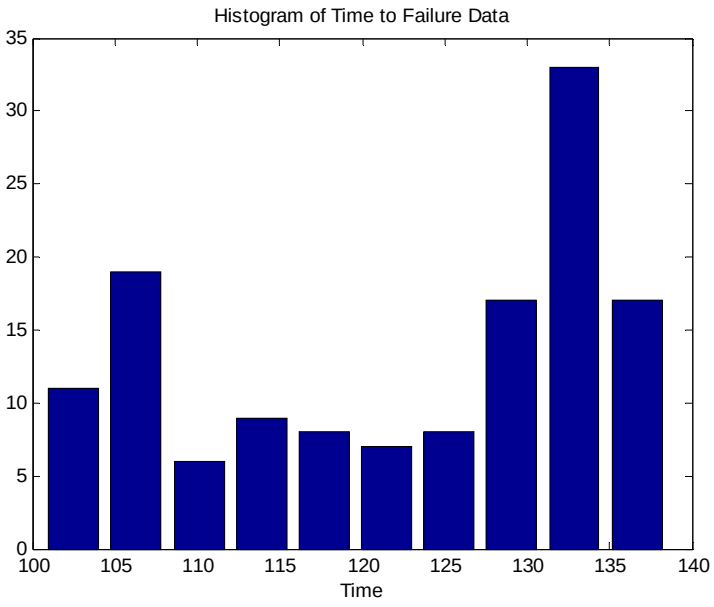


Figure 2.2-1 Histogram of Failure Times

Type I analysis begins with creating a distribution that fits the trend of the historic failure times. Four distributions, which include normal, lognormal, exponential and Weibull, are mapped to the histogram and are compared in Figure 2.2-2. The distribution that best fits the failure times will

then be used to create reliability and survival functions, these will be used to determine the RUL of this system at different system times.

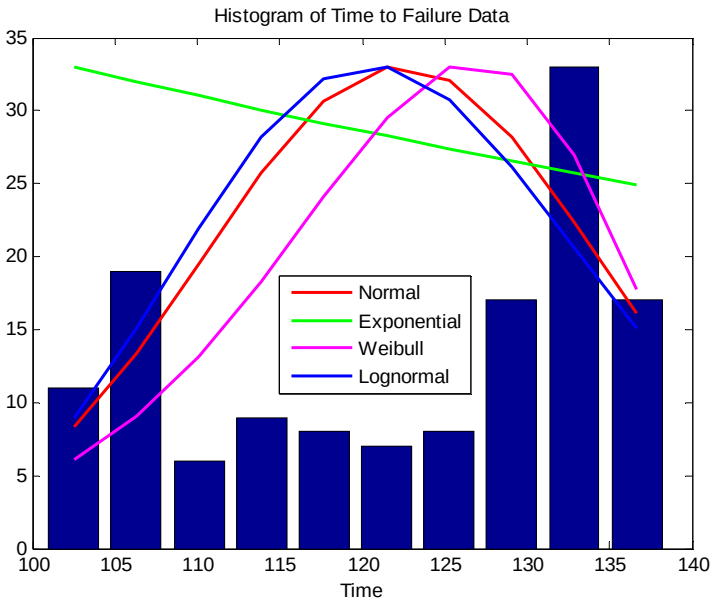


Figure 2.2-2 Histogram of Failures and Mapped Distributions

Figure 2.2-2 shows the different distribution fits that can be compared visually. Table 2-1 provides a quantitative method of comparing these distributions by comparing the calculated log-likelihood values.The distribution with the lowest value is chosen as the best fit.

Table 2-1 Distribution Log-Likelihood Values

Distribution	Log-Likelihood
Normal	525.62
Exponential	783.91
Weibull	517.75
Log Normal	528.49

According to Table 2-1, the best distribution fit is Weibull, followed closely by the normal distribution. To predict RUL it is important to know the amount of time an unfailed unit has been operating. For demonstration, Table 2-2 predicts when an average unit will fail when operating at times 0, 10, 50, 100, and 140. It is important to note that the predicted failure times are the averages of what is known. The results will be applied generally for all cases.

Table 2-2 Predicted RUL with Uncertainty

Time Run	Predicted RUL	95% Confidence Interval	
0	124.04	96.32	140.95
10	225.04	86.32	130.95
50	74.04	46.36	90.95
100	24.59	3.72	41.07
140	2.01	0.08	8.12

It should be noted that the first four RUL values differ from the fifth. The first three failure times have the RUL values centered on time 124.04, but the predicted failure time decrease as the unit survives longer. This reduction in RUL happens because when 50% of the failures are expected to occur the unit may still be operational and the RUL estimates must be adjusted by using the MRL, usually calculated by taking half the value of the expected remaining survivability. In Figure 2.2-3, the reliability function for the developed Weibull based model is shown.

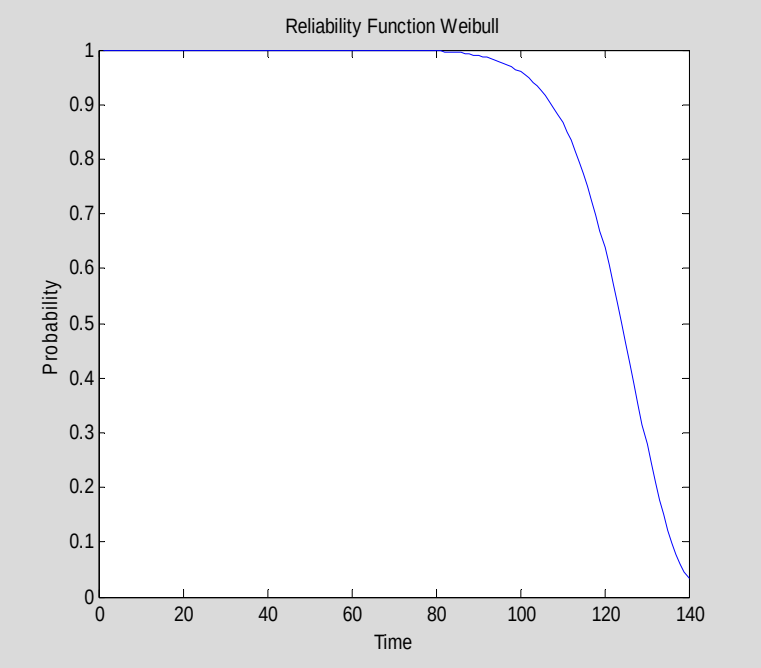


Figure 2.2-3 Weibull Reliability Curve

As can be seen in Figure 2.2-3, at time 140 the probability of failure is not zero and still has an expected RUL. The upper and lower uncertainty bounds are calculated similarly to the mean

calculation but at a different percent. The high bound is calculated at 97.5% and the low bound at 2.5% of remaining survivability.

2.2.3 Type II

Type II methods make RUL predictions for the lifetime of an average component in a specific environment. Some of the main assumptions that should be considered when using Type II prognostics are that components operating in similar conditions will degrade in a similar fashion, and that variation between units is not significant. In this demonstration, the Type II method that will be discussed and applied is the Markov Chain model.

2.2.4 Markov Chain Theory

The Markov Chain model can be applied to prognostics in two ways: condition based information can be used to predict future operating conditions of a component, and a Markov Chain can be used to create failure time predictions. The techniques necessary for this application are the creation of the transition matrix, degradation or damage calculations, and Monte Carlo simulations [Coble 2010].

The first step is identifying the possible conditions by forming the transition matrix, Q, and the initial condition probability vector, U. As mentioned before, a Markov Chain uses probabilities to predict an operating condition (OC) or state, a decision that is independent of the components current state. This is done through a transition matrix shown in **Equation 2-7**.

$$Q = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \qquad \textbf{Equation 2-7}$$

The probability P is in form P_{ij} where 'i' is the current state and 'j' is the end state. To quickly summarize the Q matrix, the probability that operating condition (OC) 1 will remain in operating condition 1 is P_{11}. The probability that OC 2 will end in OC 3 is P_{23}. This 3x3 matrix implies that there are three possible operating conditions. It is also necessary to calculate the probability of the initial starting condition. This is defined by vector **u**, which has a vector length equal to 'i' and the probability of the 'i'th unit is the probability of starting in that OC.

At each state a certain amount of damage or none at all can result. The amount of damaged received can be estimated through analysis of the data. A simple equation to calculate degradation with has the form of **Equation 2-8**.

$$\text{degradation}=d_1*t_1+d_2*t_2+d_3*t_3+\ldots+d_n*t_n \qquad \textbf{Equation 2-8}$$

In **Equation 2-8** d_i is a degradation parameter that can be estimated and t_i is the time in each state. A technique to find the parameter is to use linear regression to determine the average damage received by a component when operating in a specific OC for a full duty cycle. A duty cycle is the time spent in a specific state and is a fraction of the total operating condition time. Some assumptions of this method are that there is a linear fit to the degradation model and that there is a failure threshold, the crossing of which implies failure. When both the probability matrix and degradation estimates for a set of unfailed cases are obtained, the RUL can be estimated through Monte Carlo simulations. Using Monte Carlo simulations produces a range of estimated results that can be mapped to a distribution. To estimate the RUL of a component that has yet to fail, possible paths to failure from the last observed point are created. The result will be a distribution of failure times that inform the user of when failure is most likely to occur.

2.2.5 Markov Chain Application

To show an application of a Markov Chain model, operating condition information from 100 failed tires are used to develop a probability matrix (**Q**), degradation estimates for each operating condition, and remaining useful life probability distribution for three unfailed cases. There are three operating conditions used for these tires that pertain to road conditions. The first condition is operating on normal road conditions, the second is on off-road conditions, and the third is when the tire runs on high slip conditions. Plotting out the data from a failed tire and an unfailed tire demonstrates three distinct states, shown in Figure 2.2-4.

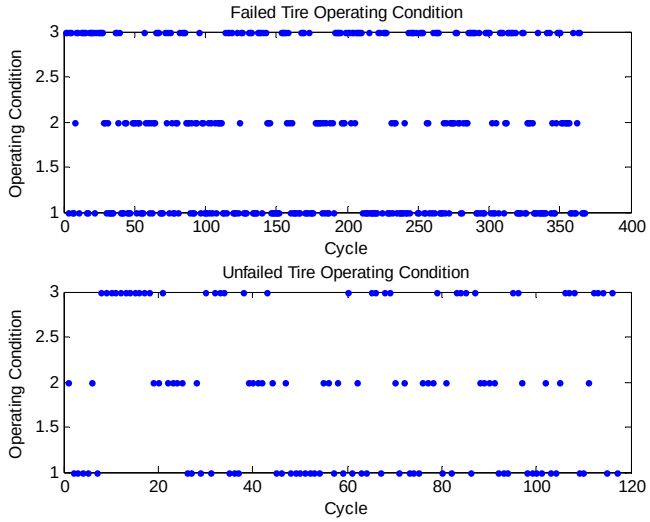


Figure 2.2-4 Failed and Unfailed Operating Conditions

The next step in developing a Markov Chain model is to find the transition probability matrix **Q**, using the failed data sets. This **Q** matrix can be populated by summing the total number of

occurrences for each possible transition, and dividing by the total number of each state transition. Table 2-3 shows the counts associated with each transition and the sum across the operating conditions.

Table 2-3 Q and u Matrices

OC	Sum	Counts			u	Q		
1	14137	7016	4371	2750	0.26	0.4962	0.3091	0.1945
2	10292	4165	4050	2077	0.55	0.4046	0.3935	0.2018
3	11963	2958	1842	7163	0.19	0.2472	0.1539	0.5987

According to the **Q** matrix values shown in the previous table, the transition with the highest probability is for OC 3 to stay in OC 3, and the least probable transition is for OC 3 to transition to OC 2. The **u** matrix represents the probabilities of each initial condition, the most probable starting condition being OC 2.

The next step in the process is determining the damage rates of each operating condition. Table 2-4 shows the damage coefficients for this set of data. These were found using linear regression and by assuming that the sum total of damage for each case based on these conditions must equal 100%.

Table 2-4 Damage Coefficients

OC	Damage Coefficient	Total Damage
1	0.1005	Mean: 99.989
2	0.2488	Std: 0.3318
3	0.5003	

After calculating the damage coefficients, a degradation graph over time was created for the failed and unfailed data sets. Figure 2.2-5 shows the failed and unfailed degradation paths in comparison with each other as well as a soft critical degradation threshold. It is important to note that the degradation threshold is normally distributed with a mean of 99.99 and standard deviation of 0.3318.

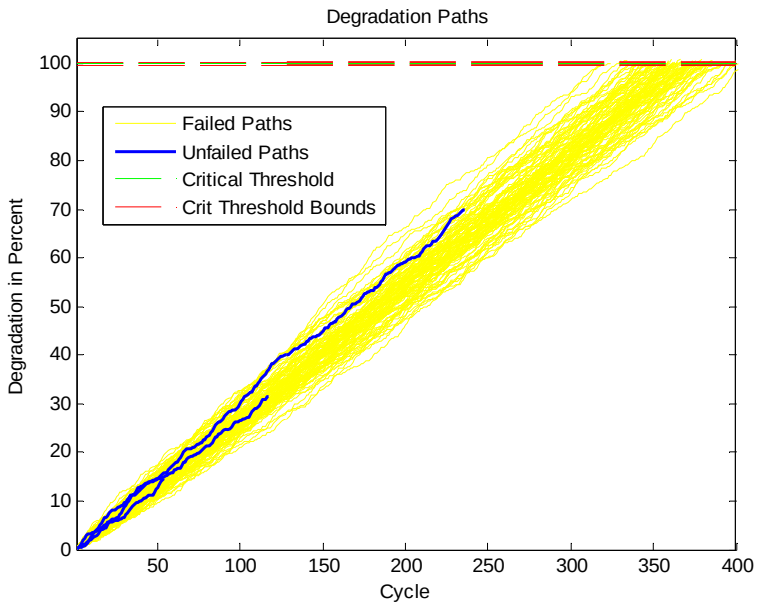


Figure 2.2-5 Degradation Plot of Tire Data with Soft Threshold

Using the **Q** matrix and the damage coefficients developed with the failed tire data sets, it is possible to create a TTF or RUL prediction.

Table 2-5 highlights the results of this process for the three separate query degradation paths, shown as the blue lines in Figure 2.2-5. Each prediction is created by first simulating 1000 possible continuing pathways for each query case. This is done by applying the appropriate amount of damage from simulated movement between conditions, then calculating the average failure time for all the simulated pathways.

Table 2-5 Unfailed Cases TTF Estimates

Unfailed Case	Time Cycles	TTF (Cycles)	Std. Dev
1	54	366.5	15.2
2	117	368.2	13.7
3	236	346.7	9.7
Population	-	365.9	17.1

Notice how cases 1 and 2 have very similar TTF estimates, while case 3 does not. For comparison, a population estimate was created using a simulation without any unfailed cases. Important to note also, is that as the known path progresses in cycles, the standard deviation of those estimates goes down. This is equivalent to a drop in the overall uncertainty for those particular cases. Figure 2.2-6 below, shows the empirical Probability Density Functions (PDF) of the predicted failure times for each of the unfailed cases and of the population estimate.

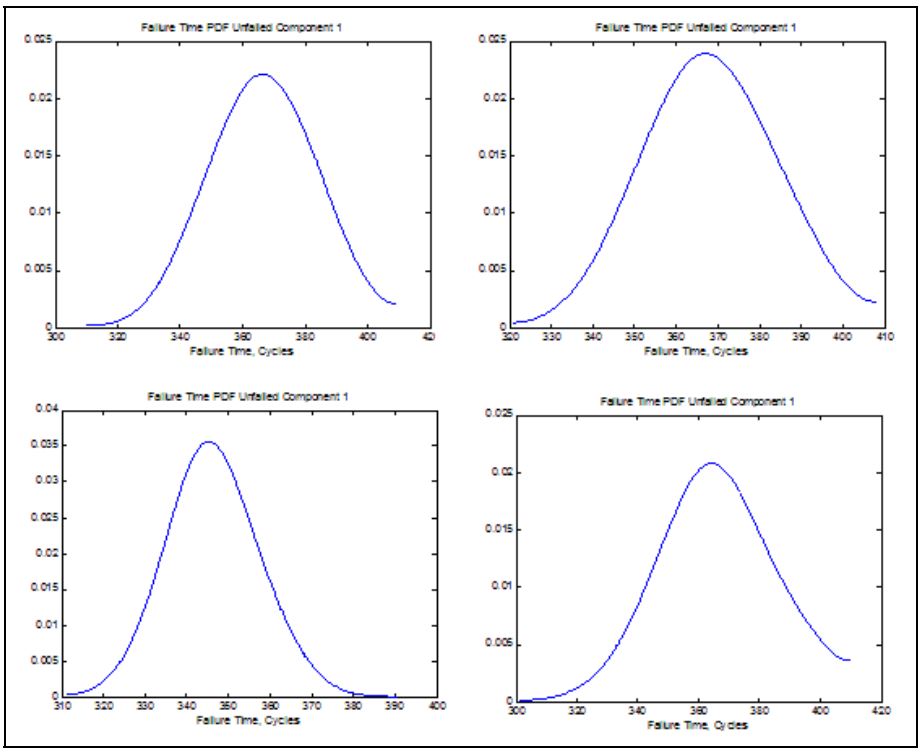


Figure 2.2-6 Distributions of Predicted Failure Times for Unfailed Cases 1:3 & Population

The distribution variance for each of these cases can be equated to the uncertainty of the estimates. These PDFs were formed using an empirical kernel smoothing technique base on the predicted failure time data, though other empirical density estimating techniques may be equally valid.

To conclude, predictions from a proof of concept Markov Chain model are used to estimate the Time to Failure (TTF) or RUL of tires when only the past and current operating condition data is known. A Markov Chain model is a memory-less process of states, or operating conditions, where the probability of moving from one state to the next is known and takes the form of the **Q** matrix. Using failed tire data with three distinct operating conditions, a probability matrix, **Q**, was developed for transitioning between operating conditions and vector **u** was found for predicting starting condition position. Using the same failed tire data, damage estimates were also created

for each operating condition. Finally the probability matrix **Q** and the damage estimates were combined to form a predictive model that was simulated 1000 times for each unfailed case producing three RUL distributions. A soft degradation threshold was also used based on the calculated damage values.

The uncertainty distributions for the unfailed cases 1, 2, and the population estimate were very similar, being close in both mean and standard deviation. Case 3 though, had a slightly different mean, roughly 6% smaller than the other three estimates. In fact, case 3 exists on the edge of case 1 and 2's failure distributions. This difference in TTF estimates is likely due to the fact the the majority of the observed paths were in lower damage states.

For a more complicated system (a system with more operating conditions) it would be fairly simple to create a larger **Q** matrix. Using a Markov Chain to predict failure would be more difficult if no failure was observed, if system repair needed to be accounted for, or if there were state changes inconsistent in time. Using Monte Carlo allows for easy estimation of uncertainty.

2.3 Type III

Type III predictions of RUL and TTF provide estimates specifically for the individual component being tested. While Type I generalizes information about a components lifetime, and Type II generalizes information about a component within a known environment and/or operating condition, Type III tailors RUL predictions based on sensed signal information taken from a particular unit that relate directly or indirectly to degradation. This takes into account not only the individual components' operating condition and environment, but also some quantitative value of wear, which can be trended and predicted using techniques such as GPM.

Discussed below are the approaches for developing this type of model and comparing three types of fittings: linear, quadratic, and exponential. Parameters of the equations are approximated from linear and non-linear regression. The section will later connect model parameters with prediction uncertainty.

2.3.1 Type III GPM Model Types

The equations listed in this section are meant to describe the degradation process by fitting it to some known functional form. The first fitting will be a linear model shown in **Equation 2-9**.

$$y = \beta_1 x + \beta_0 \qquad \textbf{Equation 2-9}$$

In **Equation 2-9** y describes the degradation path, and β_1 and β_0 are the parameters. With this fitting, the parameter β must be found to form a prediction of y. Next, **Equation 2-10** is a quadratic model. In this equation there are three parameters that need to be solved for.

$$y = \beta_2 x^2 + \beta_1 x + \beta_0 \qquad \textbf{Equation 2-10}$$

Finally there is an exponential model shown in **Equation 2-11**.

$$y = \beta_2 e^{\beta_1} + \beta_0 \qquad \textbf{Equation 2-11}$$

Using the exponential functional form it is also possible to linearly transform the paths using a natural log and solving for the parameters as one would for a linear function.

2.3.2 Type III GPM Model Methodologies

The three different GPM architectures presented in this section are functions that describe common trends in the evolution of degradation in many systems. Lu and Meeker first formally described the GPM and some of its applications towards crack growth in 1996. Figure 2.3-1 illustrates some of the sources of uncertainty associated with any GPM.

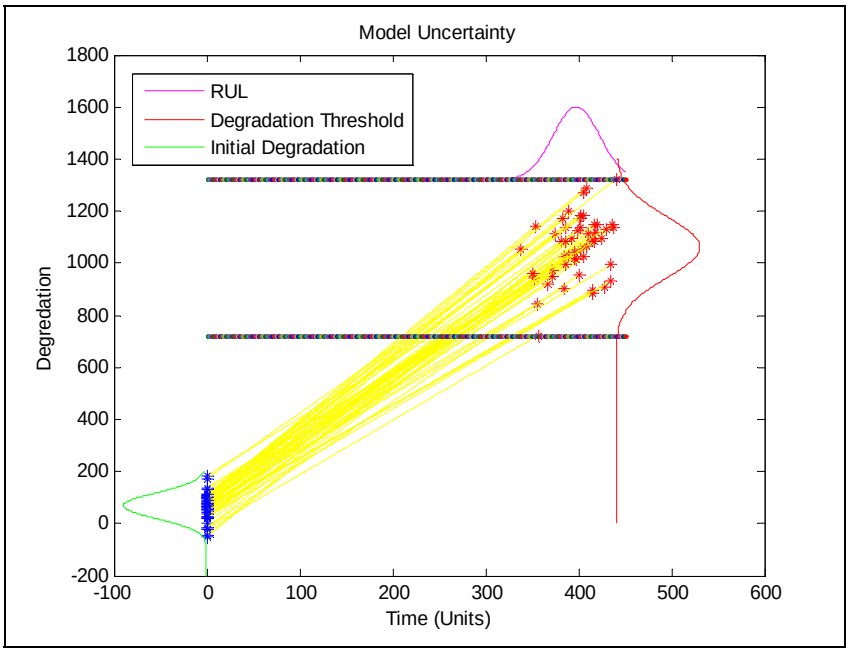


Figure 2.3-1 Linear General Path Model Characteristics

Each yellow line represents simulated component degradation paths from beginning of life to failure. The starting points of each line are represented by blue asterisks at time zero, and the

failure points are shown as the red asterisks. There are three PDFs that represent three basic types of uncertainty that can be visualized in this figure. The purple horizontal PDF represents variation in the Time of Failures (TOF). This is the actual distribution of failure times, and can be used to determine the accuracy of the predicted RUL values. The red PDF, labeled as the degradation threshold, represents the variation of degradation levels for the simulated components. Lastly, the green PDF represents the initial degradation levels, which could be attributed to manufacturing or installment differences. This distribution quantifies the variation in the beginning levels of degradation for a component. These, along with model uncertainty and any parameter regression uncertainty, are the basic sources of uncertainty for a GPM.

2.3.3 Model Uncertainty

As previously mentioned, there are multiple sources of uncertainty that are important to consider when applying the GPM for system prognostics. The sources covered within this section are the variability in the initial degradation amount, the uncertainty of the regressed model parameters, the measurement uncertainty, and the failure threshold uncertainty. The selection of a model type will also contribute to the uncertainty of the RUL prediction. If a linear model is used to determine degradation paths that are non-linear in nature, a model bias will occur and the uncertainty of the RUL estimate will be larger. The sources of uncertainty are analyzed using the three model fits previously discussed. The increase in degradation over time is assumed to be linear in parameters, linearly transformable and be fit to the three functional forms.

Initial Degradation Uncertainty

Initial degradation is the starting degradation level at time zero. While initial degradation is usually a minor form of uncertainty in a linear path model, it does influence the uncertainty in the RUL estimate. In **Equation 2-9** it is shown that β_0, the y-intercept, is the initial degradation component; therefore controlling the distribution of the initial degradation is a matter of setting the distribution of β_0 in the model. In most uses though, the initial degradation parameter is constant and starts at zero, but if the effects of the initial uncertainty are substantial, one can map the initial degradation points to a normal distribution. This method is demonstrated in Lu and Meeker 1996. In practical applications, not all components are created equal. Errors and flaws in the manufacturing process can cause components to begin life at varying degrees of degradation; for example, the initial tread depth of a tire has some manufacturing variability. This variability in the manufacturing process can lead to initial variability in the degradation model.

Prior Uncertainty

Prior information or knowledge can be used with Bayesian updating to develop better parameters, especially near the beginning of life. There are many methods for producing priors with a degradation model. Examples of methods for forming priors can be found in Usynin 2007. In Gabraeel and Elwany 2009, prior distributions are formed using the Bernstein Distribution, which is a gamma distribution of the RUL uncertainty when a linear path model is used with a hard degradation limit. Gutierrez-Pulido and Aguirre-Torres 2005 provide a more common and easily applicable method for forming prior distributions. Other sections provide further details of regression using Bayesian statistics. In their implementation, the models used are very general and have at most two parameters. Prior information comes in the form of mean and standard deviation of time-to-failures or as a quantile of time to failures.

The priors used in the model for Bayesian methods are formed from model regression. As more data accumulates during component monitoring, the weights of the priors are decreased and the weighting of the data is increased. This was demonstrated by Usynin 2007. In other words, as more observed data becomes available, the RUL predictions become more certain. This is demonstrated in Figure 2.3-2 and Figure 2.3-3. It is easy to notice that in Figure 2.3-2 the Ordinary Least Squares (OLS) prediction is highly skewed away from an accurate prediction and has a very high uncertainty (blue shaded cone), while the Bayesian estimate is much more certain (brown shaded cone). In Figure 2.3-3 however, as more information becomes available the predictions both become more accurate and focused. The advantage of Bayesian methods that use prior information is an effective prediction early in life.

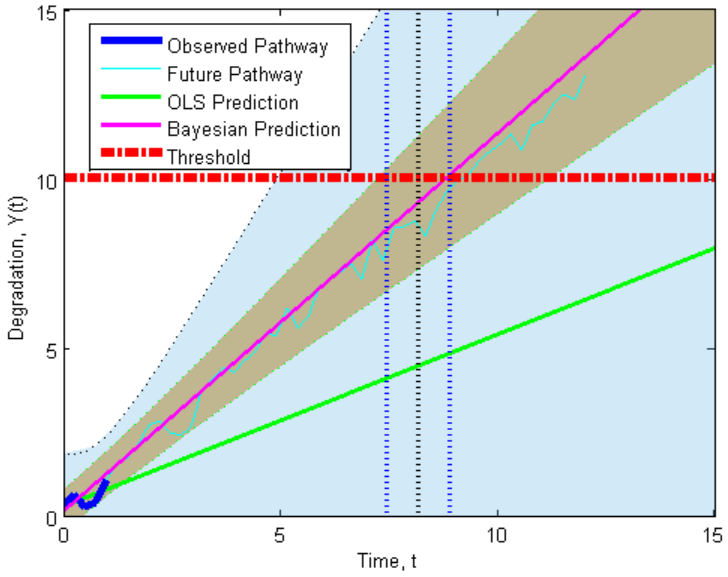


Figure 2.3-2 Model Uncertainty near Beginning of Life

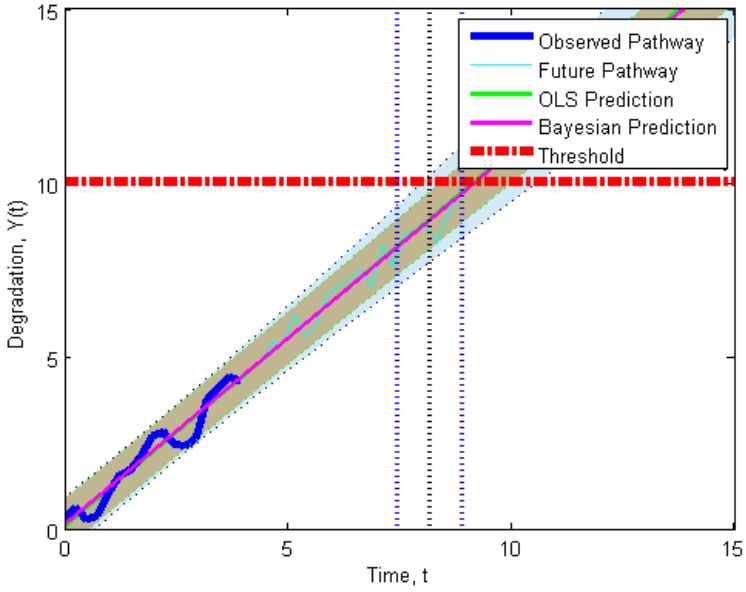


Figure 2.3-3 Model Uncertainty Later in Life

Measurement Uncertainty

Measurement uncertainty is a major source of uncertainty in model parameters and originates in the data collection process. Jie, Yun, and Wiesel (2008), demonstrate a method to simulate measurement data with Gaussian uncertainty. The measurement uncertainty used in this analysis

was created using a normally distributed stochastic quantity of the form of $N(\mu,\sigma^2)$, added to a linear function with the parameters $\beta 1=0.2$, and $\beta_2=1.2$. To demonstrate that an increase in measurement uncertainty increases the prediction uncertainty, Bayesian linear regression and OLS were applied at two different measurement noise levels. The difference in RUL uncertainties given increasing measurement uncertainty is presented graphically in Figure 2.3-4.and Figure 2.3-5.

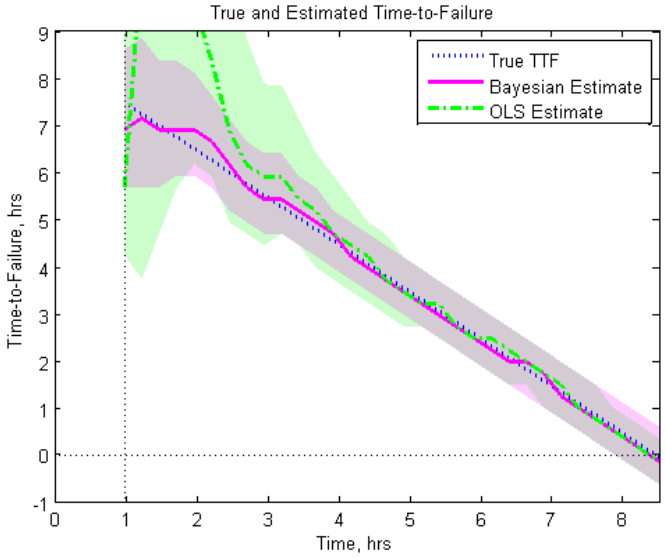


Figure 2.3-4 Test and Estimated Time-to-Failure

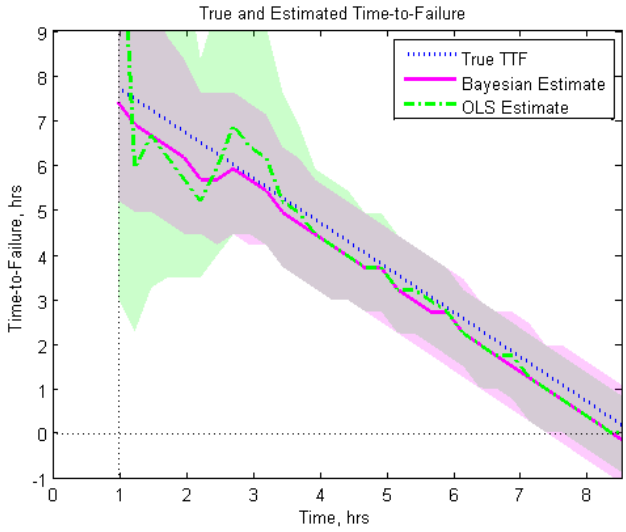


Figure 2.3-5 True and Estimated Time-to-Failure with Increased Measurement Uncertainty

The shaded lines represent the confidence limits, matching in color with the estimation method. Clearly the uncertainty in Figure 2.3-4 is less than the uncertainty in Figure 2.3-5, demonstrating that a higher measurement uncertainty leads to an increased uncertainty in the RUL estimate.

Degradation Threshold Uncertainty

Previously, the developed model used a critical degradation threshold that acted as a hard limit of failure. Figure 2.3-1 details a linear path model that has each component failing at varying degradation levels, this model .has no implicit degradation threshold. The threshold was quantified by creating a normalized PDF of the failure degradation levels. The PDF forms the basis of the new degradation threshold.

Figure 2.3-6 shows how the model uncertainty and the threshold uncertainty are combined to form a prediction of the RUL distribution. By first using OLS regression to determine the model parameters and their associated uncertainties, a Monte Carlo style simulation can use these in conjunction with the failure point distribution to find not only the most probable TTF, but also a quantifiable measure of uncertainty. For the case below, let the two regressed linear parameters be b1, and b2 with an associated confidence level of + or – E1 and E2 respectively. These are used as the mean and variance of two normally distributed, random sampling distributions of each parameter. By repeatedly randomly sampling a parameter from each of these distributions (B_1, and B_2), and also one from the failure point distribution (Y), a distribution of failure times can be formed by solving the linear equation ($Y = B_1 + B_2x$) for each set of samples parameters. This distribution provides not only a most probable time to failure, but also the associated uncertainty.

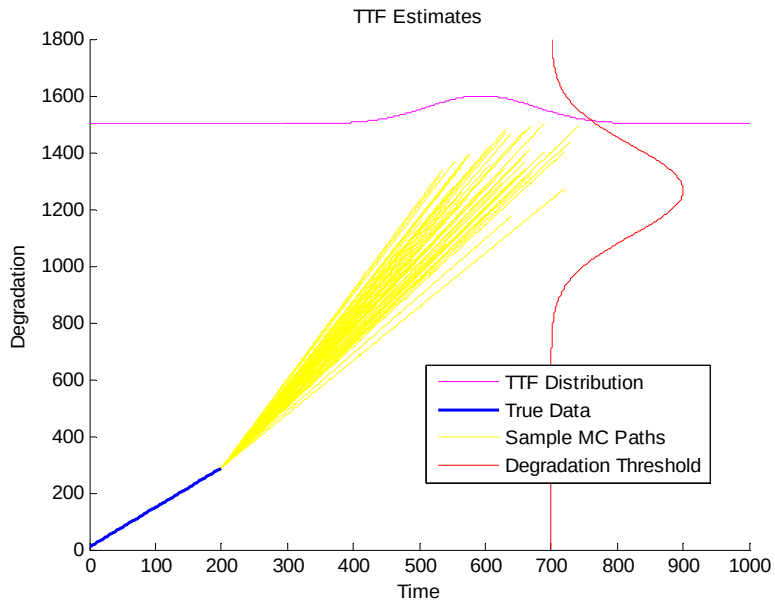


Figure 2.3-6 Soft Degradation Threshold Monte Carlo Estimate

Figure 2.3-6 shows an average fit degradation path and its predicted RUL distribution, set at an average of the degradation distribution developed using the prior population data. The predicted RUL estimates form a normal distribution denoted in the horizontal magenta distribution. The Monte Carlo paths show degradation predictions, and the parameter distribution shows the distribution of the new data set.

The lessons learned pertaining to linear degradation model uncertainty can be applied to non-linear degradation model uncertainty. This includes linear regression of non-linear data, and non-linear regression of non-linear data as well as quadratic and exponential models.

2.3.4 Model Transformations

Since uncertainty has been discussed, this section will focus on model types and algorithm predictions. When it comes to estimating the RUL of components with non-linear degradation paths, linear models will lead to less than ideal results. However, linear tools and algorithms can be directly applied and have straight forward uncertainty quantification methods. Therefore, it is of interest to use the flexibility of non-linear models with the tractability and robustness of linear models. Non-linear models may require the use of a technique known as linear transformation to put them in a form where linear techniques can be applied directly. One purpose of linear model transformation is to create the linear relationships between variables. The correlation between x and y should not change after the transformation because this would imply that non-linearity exists between x and y.

Linear in Parameters

Linear models are the most widely used empirical models. The term "linear" refers not to the shape of the path, but the linear addition of parameters. Theoretically functions such as sin(x), e^x, and x^2 can be included in linear models, as long as those terms have an additive effect on the response y.

Quadratic Transformation

To transform a quadratic, the response y is square rooted. The process is described in the **Equation 2-12**. The model for the GPM has normally distributed parameters with mean and variance and a stochastic noise value.

$$\sqrt{y} = \sqrt{\beta_1 x^2 + \beta_0 + \varepsilon} \qquad \textbf{Equation 2-12}$$

The model and the transformed model are shown in Figure 2.3-7. The quadratic formula defines the left side of the figure and **Equation 2-12** defines the path model of the right side. There are 500 paths that are shown in each graph. It is important to note that when the square root of the paths was taken, the square root of the critical degradation threshold was used as the new critical degradation threshold for the transformed model. Transforming the quadratic model is not necessarily needed to estimate the RUL of the simulated data.

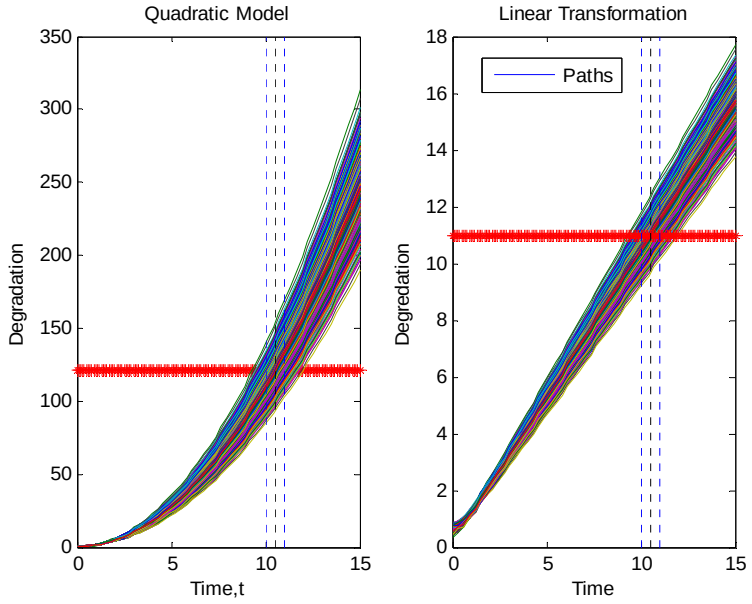


Figure 2.3-7 Quadratic Model and Transformation

Logarithmic Transformation

The exponential model has been defined previously, Figure 2.3-5 can be linear in parameters, depending on the model, but due to its non-linear nature, using a transformation is easier to analyze. The transformation used is described in **Equation 2-13**.

$$\ln(y) = \ln(\exp(\beta_1 x + \beta_0) + \varepsilon) \qquad \textbf{Equation 2-13}$$

The parameters in this equation are normally distributed with a mean and variance. There is also a stochastic noise value with a mean and variance. The critical degradation threshold for the untransformed GPM is e^{11}. For the transformed GPM, the critical degradation threshold was also transformed in the same way, setting the critical threshold at 11. Exponential transformation is shown in Figure 2.3-8.

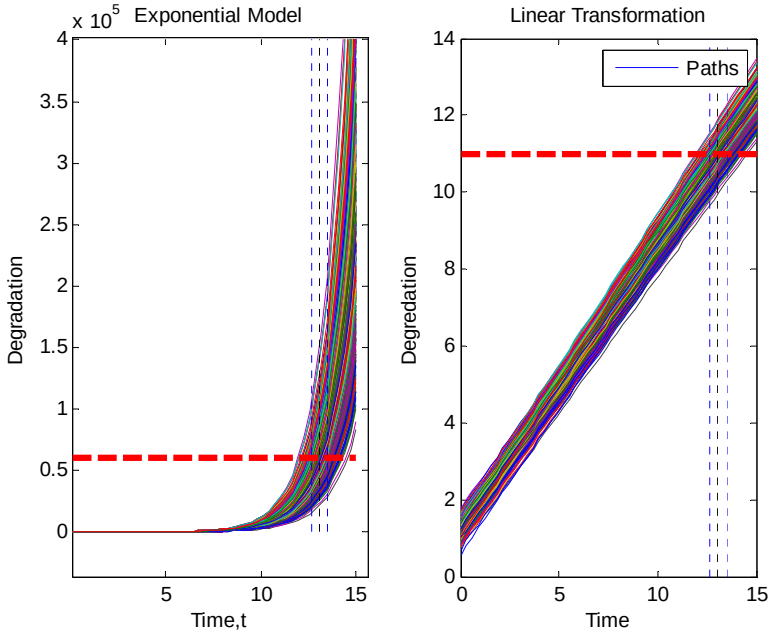


Figure 2.3-8 Exponential Model and Transformation

The exponential model, post transformation, appears to be linear and is more evenly distributed, as the shape on the right side of Figure 2.3-8 is not as triangular as Figure 2.3-7. Calculating the RUL of the transformed data using a linear model produces predictions seen in Figure 2.3-9 and Figure 2.3-10. While it may seem that the transformed exponential model is the most certain model type demonstrated, it is not necessarily the case in reality. The data used is simulated and created ideally. This section on exponential transformation demonstrates how a non-linear problem can be solved linearly.

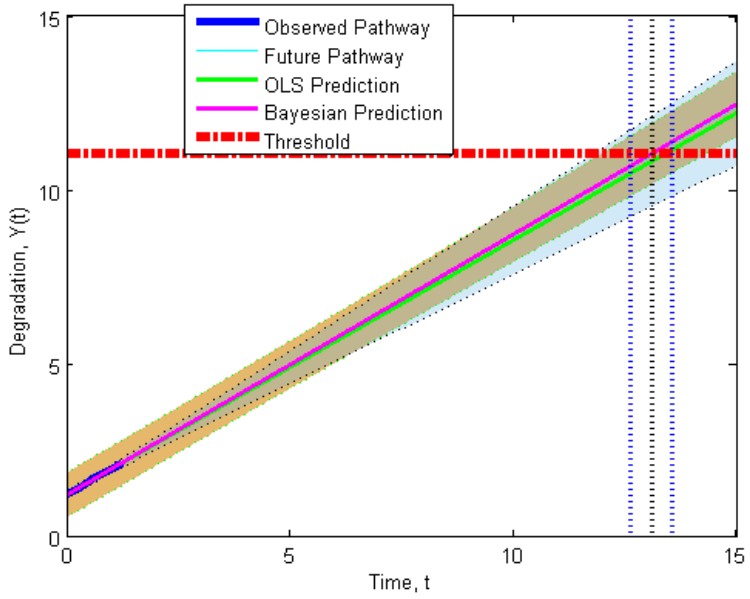


Figure 2.3-9 Exponential Transformation Beginning of Life RUL Estimates

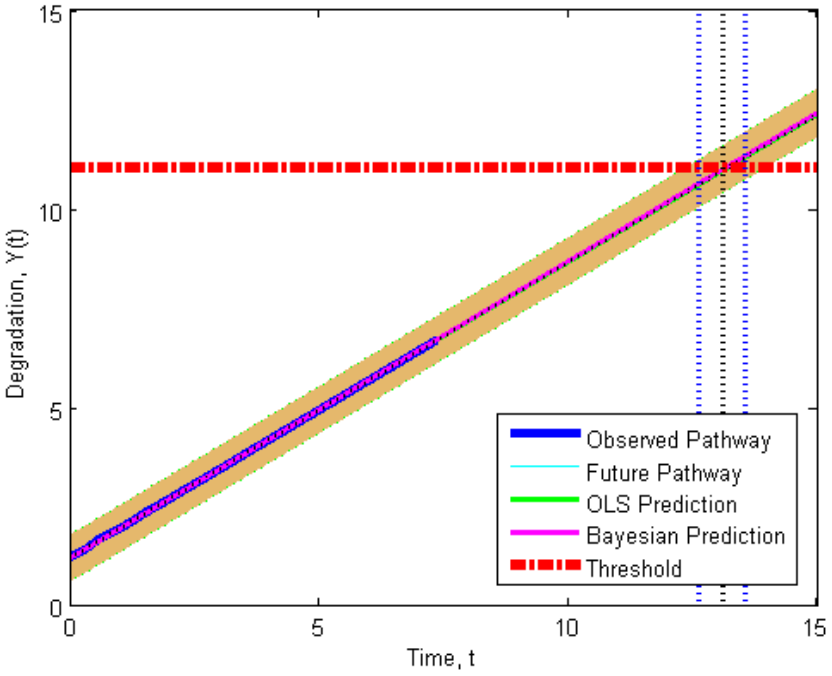


Figure 2.3-10 Exponential Transformation Middle of Life RUL Estimates

2.4 Non-Linear Uncertainty

With non-linear models such as quadratic or exponential, a method using squared error multiplied by the t-distribution is used to produce 95% confidence intervals. To demonstrate this method, simulated data, shown in Figure 2.4-1, is used to produce uncertainty calculations for quadratic and exponential regression. There are 500 simulated paths shown in green with a critical degradation threshold set at 50. Any simulated data past this threshold is extraneous.

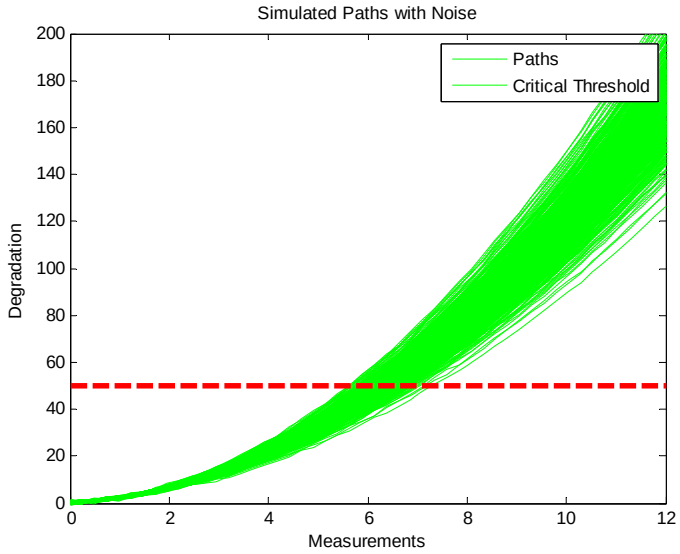


Figure 2.4-1 Quadratic Simulated Paths

For the uncertainty in these calculations, the intervals are the result of the squared error times the appropriate student t-distribution. The process requires regression coefficients beta, residuals r, and estimated coefficient covariance matrix SIGMA and is detailed in **Equation 2-14**.

$$SE = \sqrt{diag(SIGMA)}$$

$$Int = SE * student_{t(1-alpha,v)}$$

$$CI_{hi}(:) = (beta(:) + int)$$

$$CI_{lo}(:) = (beta(:) - int)$$

Equation 2-14

The results of these calculations are that each of the parameter estimates will have confidence interval estimates as well. Using quadratic, exponential and Bayesian regression on a single randomly chosen path results in the three plots shown in Figure 2.4-2 below.

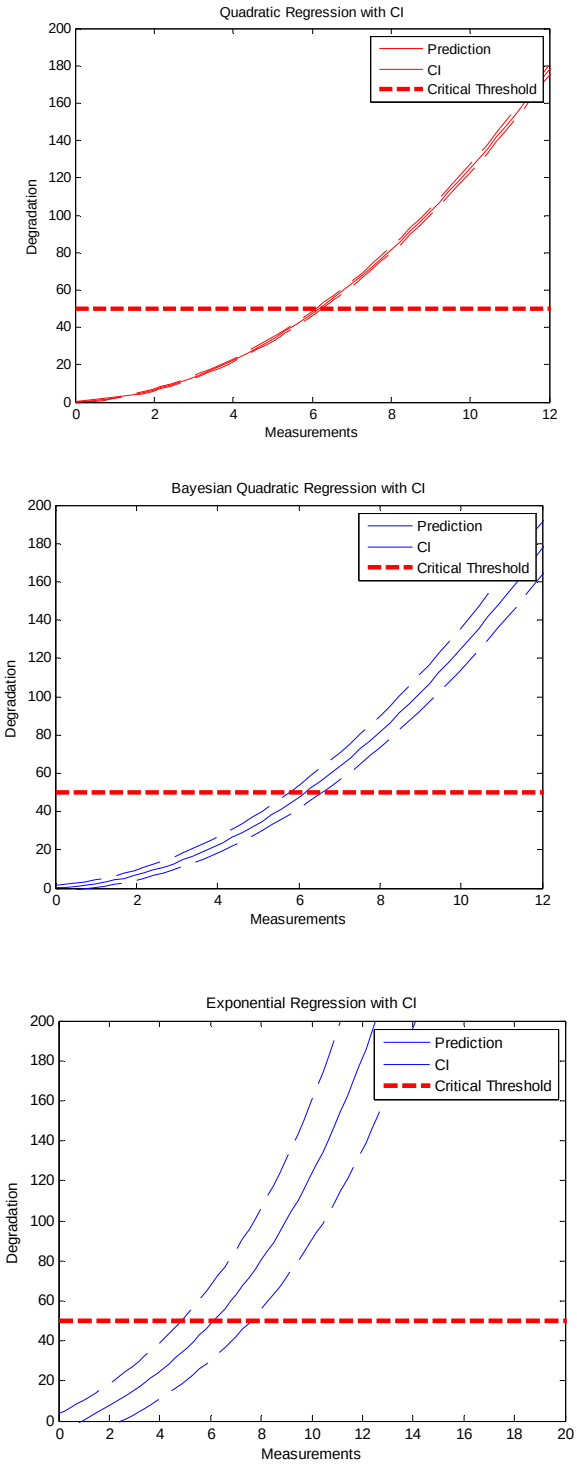


Figure 2.4-2 Quadratic and Exponential Predictions with 95% CI

Since a quadratic model defines the simulated data, the quadratic regression on the top left of the figure has the smallest CI. The exponential fit is the least suited and has the largest CI. The

Bayesian quadratic regression appears to be worse than the simple regression. This is most likely due to how the Bayesian prior information added to the quadratic regression model is not informative, or due to a lack of noise. Residuals are shown in Figure 2.4-3.

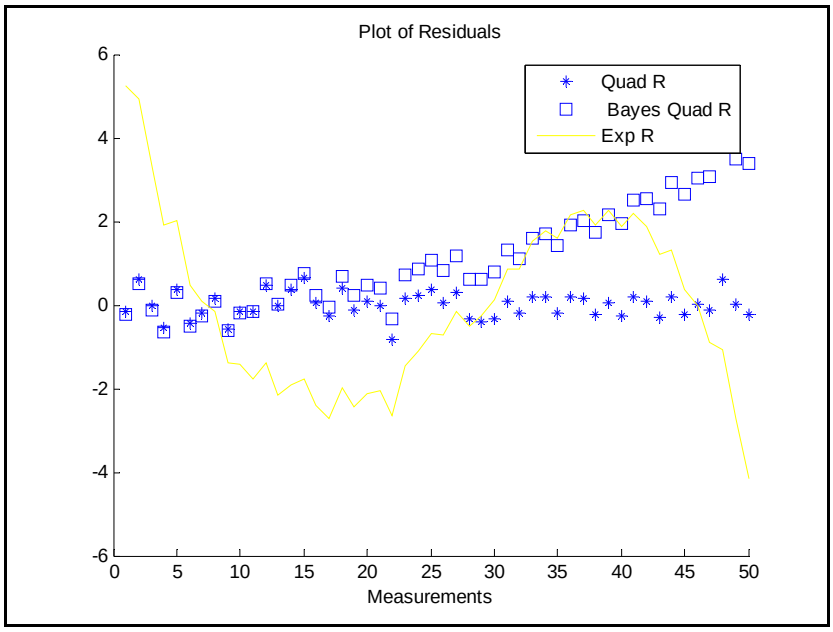


Figure 2.4-3 Prediction Residuals

The quadratic regression is the best fit, and the residuals appear to be centered on zero in the figure. The residuals are not zero due to the normally distributed noise. The size of the confidence interval appears to correlate with the variation of the residuals. The first half of the Bayesian quadratic residuals match the quadratics residuals but deviates, and the exponential residuals seem unrelated to either of the other two fits.

Solving for the model as more data is added demonstrates how the confidence intervals change over time. Figure 2.4-4 and Figure 2.4-5 shows a quadratic and exponential prediction and the associating confidence intervals. Notice how the confidence intervals cannot be calculated for the entirety of the data. This is due to an over parameterization of the developed predictive model. Essentially the predictive model needs more data before confidence intervals can be calculated.

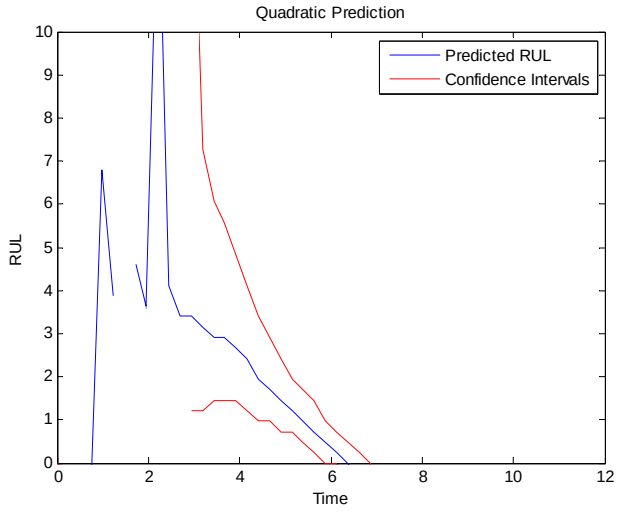


Figure 2.4-4 Quadratic RUL Prediction

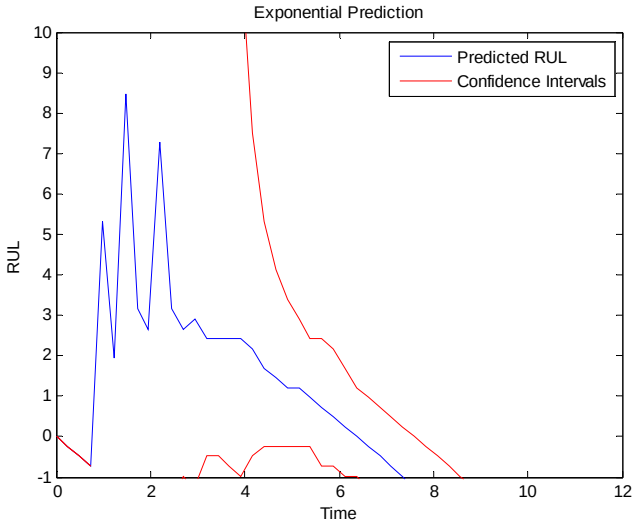
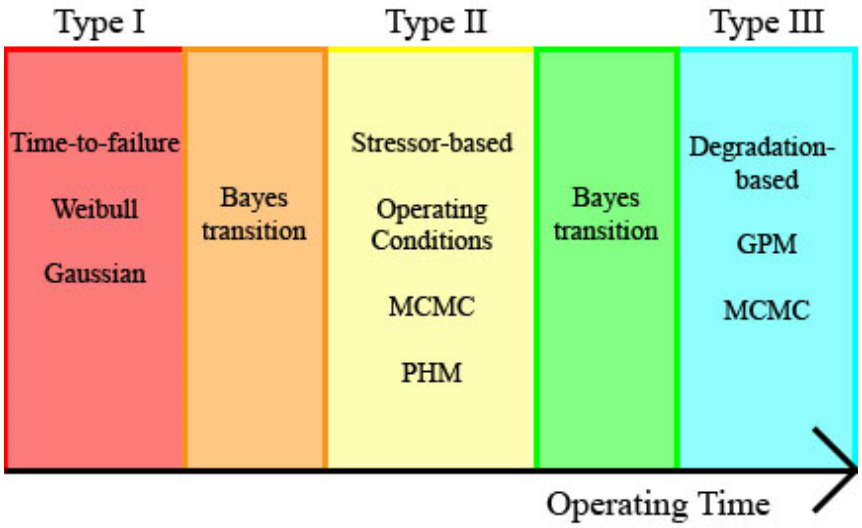


Figure 2.4-5 Exponential RUL prediction

Comparing the two figures shows that the exponential model holds more uncertainty than the quadratic comparison for this simulation. Since the simulated data is based off of a quadratic model with Gaussian noise it makes sense that the exponential model will contain more uncertainty in its prediction than a quadratic model would.

3. Task 2: Lifecycle Prognostics

Each prognostic type uses different information and requires different algorithms and models. The data, and thus type, can be categorized chronologically to the component's life. Prior to operations the most only data available about a system would be how previous systems have acted under similar conditions, this would be labeled Type I information. Type II information incorporates the known and expected operational and /or environmental stressors associated with the system. The final class of information, Type III, is derived from directly observed signals and sensors attached to the system. This task seeks to establish a unified, meaningful, and logical progression through utilizing the various types of information as they become available within the operational lifetime of a system.



Error! Reference source not found.**. Lifecycle Prognostics with Bayesian Transitions**

Population based (Type I) models would typically be used before the system operates and during the initial phases right after startup. Stressor based models (Type II) are most accurately employed starting after some expectations of the current and ongoing operational stresses of the system have been established, but also typically before any initial indications of a fault or any incipient degradation is detected. The final phase of modeling, a Type III online effects based model can be used to predict propagation of some negative health indicator after its existence has been established by some detection model. Developing a method of transitioning between these model types in a manner that best preserves and incorporates information and knowledge gained

from previous stages of the lifetime modeling maximizes the use and presentation of relevant information about a system. This aids in providing the most accurate and robust indication of a system's reliability or expected RUL at any point in the system's operational lifetime. The goal of this task is to develop such a transitioning method.

As is indicated in Figure 3.1-1, one approach investigated is to apply forms of Bayesian transitions between the primary types of information and models. These methods combine previous estimates of a parameter (priors) with sampling data to produce a posterior estimate that combines both sources of data, while reducing uncertainty. Within the context of Lifecycle Prognostics, Bayesian transitioning methods can be applied to bridge between prognostics model and information types.

Additional methods beyond Bayesian Transitions have also been explored during this investigation to come up with consistent RUL estimates when employing the GPM. These typically have involved weighted incorporations of the Type I and Type III estimates. As the component moves forward in time, the weights shift from favoring Type I to favoring Type III predictions. These investigations are discussed in the following section and lead to the development of an regimented method for incorporating information and transitioning between the types of prognostic models.

3.1 Bayesian and Classical Statistics

The classical form of Bayesian statistics is based primarily on Bayes' formula (Ghosh 2006).

$$\pi(\theta|x) = \frac{\pi(\theta)f(x|\theta)}{\int_{\Theta}\pi(\theta')f(x|\theta')d\theta'}$$

Equation 3-1

Equation **3-1** calculates the conditional probability density function (PDF) of θ, the parameters of interest, given new data x. The prior density function, $\pi(\theta)$, is a prior belief or estimate, with some characteristic distribution about the parameters. The $f(x|\theta)$ is the density of x, the data, interpreted as the conditional density of x given θ. The numerator is the joint density of θ and x, while the denominator is the marginal density of x, or prior predictive distribution. It is considered prior because it does not depend on the data x, and predictive because it describes a quantity that is observable. If the parameter of interest is discrete, then the integral is replaced with a summation. Both the sum and integral are integrated over all possible values of θ. Equation **3-1**, when solved, gives the posterior density, a quantification of uncertainty about θ in

light of new data x. The transition from $\pi(\theta)$ to $\pi(\theta|x)$ is what is learned from the data, changing the mean estimate and uncertainty.

3.1.1 Gaussian Conjugate Distributions

Computationally, the posterior calculation of Equation **3-1** could be mathematically difficult for every situation. However there exists prior and posterior distributions of the same family, jointly called conjugate distributions, which simplify Bayesian calculations greatly. The Gaussian conjugate distribution not only has Gaussian prior and posterior distributions, but also a Gaussian data sample. First the Gaussian prior distribution is defined as

$$\pi(\mu,\tau) = \frac{1}{\sqrt{2\pi\tau^2}}\exp\left(\frac{-(\mu-\eta)^2}{2\tau^2}\right)$$

Equation 3-2

Let $X_1, X_2 \ldots X_n$ be independently and identically distributed (i.i.d.) sampling data ~ $N(\mu, \sigma^2)$.

$$f(x|\mu) = \prod_{i=1}^{n} f(x_i|\mu) = (2\pi\sigma^2)^{-n/2}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right)$$

Equation 3-3

These equations are combined with Equation **3-1** to yield

$$E(\mu|x) = \frac{\frac{\eta}{\tau^2}+\frac{\sum x_i}{\sigma^2}}{\frac{1}{\tau^2}+\frac{n}{\sigma^2}} = \frac{\frac{\eta}{\tau^2}+\frac{n}{\sigma^2}\bar{X}}{\frac{1}{\tau^2}+\frac{n}{\sigma^2}}$$

Equation 3-4

$$var(\mu|x) = \frac{1}{\frac{1}{\tau^2}+\frac{n}{\sigma^2}}$$

Equation 3-5

Equation **3-4** gives the expected, or most likely, estimate of the mean of the distribution, while Equation **3-5** gives the variance of the mean. It is important to distinguish between variance of the distribution, and variance of the mean. This variance is a measure of certainty of the expected mean estimate. The smaller the variance, the more precise the estimate is.

The prior variance τ^2 measures the strength of the belief in the uncertainty of the prior distribution. In this sense $1/\tau^2$ is the precision of the prior, while n/σ^2 is the precision of n data points (Welch 1939). This means that the posterior mean is the weighted average of the prior and

sample means with the precisions of each as weights. It also means that with more data, the prior data is weighted less and eventually the most likely estimate approaches the posterior.

3.1.2 Weibull Sampling Distribution

Another useful pair of conjugates is the inverse gamma, which is used when the sampling data most closely fits a Weibull distribution, assuming a known shape parameter β. In this case the prior inverse gamma parameters are defined as "a" and "b". If n numbers of i.i.d. sample points X fit a Weibull, the posterior is given by Equation **3-6**.

$$\pi(a, b|X) = a + n, b + \sum_{i=1}^{n} x_i^{\beta} \qquad \textbf{Equation 3-6}$$

Several more conjugate pairs can easily be referenced.

3.1.3 OLS Regression

Using ordinary least squares (OLS) regression, a linear model can be defined by the general form, **Equation 3-7**, where Y is the response, X is the input matrix, β is the vector of parameters, and $\sigma^2 I$ represents independent observation errors with equal variance. OLS assumes that the errors are normally distributed around a zero mean. The solution to the parameters can be found by solving the least squares solution, **Equation 3-8**. The columns of X are the independent parameter measurements and X includes a column of ones to allow for a non-zero intercept.

$$Y|\beta, X \sim N(X\beta, \sigma^2 I) \qquad \textbf{Equation 3-7}$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y \qquad \textbf{Equation 3-8}$$

3.1.4 Linear Regression with Bayesian Priors

As linear regression is one of the most widely used statistical tools, it makes sense to apply Bayesian analysis to develop more sophisticated models. For the Bayesian OLS model in this report, because the noise variance of Y, σ^2, is known, the conditional posterior distribution of β given σ^2 is Gaussian (Gelman 2004), for which the conjugate prior distribution also takes on a Gaussian form. The conditional posterior distribution for the parameters is then **Equation 3-8**, combined with **Equation 3-9** and **Equation 3-10**.

$$\beta|\sigma^2, y \sim N(\beta^*, V\sigma^2) \qquad \textbf{Equation 3-9}$$

$$V = (X^T X)^{-1}$$ **Equation 3-10**

Before the Bayes prior information is incorporated, a data covariance matrix Σ is introduced. Instead of assuming equally distributed errors, $\sigma^2 I$, the covariance matrix is an **n** x **n** symmetric positive matrix, containing the variance at each point. **Equation 3-8** and **Equation 3-10** are then replaced by **Equation 3-11** and **Equation 3-12**.

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y$$ **Equation 3-11**

$$V_{\hat{\beta}} = (X^T \Sigma^{-1} X)^{-1}$$ **Equation 3-12**

To include Bayesian updating, the prior distribution $\sim N(b_0, \Sigma_b)$ is treated as one additional data point to the OLS input matrix. To achieve this, each variable is appended with the prior distribution data, **Equation 3-13**. The X is appended with an identity matrix, with ones representing the parameters for which prior distributions exist.

$$y_a = \begin{bmatrix} y \\ b_0 \end{bmatrix}, X_a = \begin{bmatrix} X \\ I_k \end{bmatrix}, \Sigma_a = \begin{bmatrix} \Sigma_y & 0 \\ 0 & \Sigma_b \end{bmatrix}$$ **Equation 3-13**

Two pieces of information, the prior and the data, are used to form an estimate which is the posterior. The weightings of these two pieces of information are dependent on the variance of the prior, the variance (uncertainty) of the data, and the amount of the measured data (number of samples). If the variance of the prior is small compared to the uncertainty of the data, the prior b_0 will be weighed more heavily. However, as more data is collected, the data instead of the prior will be weighted more heavily in calculating the posterior.

3.1.5 Local Linear Regression and Locally Weighted Regression

The OLS model is sometimes referred to as "global" when in contrast to Local Linear Regression (LLR) and Locally Weighted Regression (LWR). As implied by the names, both LLR and LWR are used to correlate portions, or windows, of data as opposed to the entire dataset. For example, if the window size (bandwidth) were 10, the LLR and LWR would take in 10 data points and form a regression model. For LLR the standard OLS model, Equation **3-7** through Equation **3-13**, apply.

For the LWR model, the dataset is weighted based on the proximity to some arbitrary point in the dataset. Though there are many different kinds of LWR models, this investigation focuses on the one-sided Gaussian kernel:

$$w_j = \frac{\exp\left(-(X_{j0} - X_j)\right)^2}{bw^2}$$

Equation 3-14

where X_{j0} is the center of the kernel, X_j is the vector of dependent variables including X_{j0}, and bw is the bandwidth, to yield for each data point the weight w_j. When applied the center of the kernel is given the maximum normalized weighting of 1, with further points receiving diminishing values. The bandwidth determines the rate of decreasing weights. When applying local models, the bandwidth is an important parameter to establish. There is a tradeoff between weighting the ending trends of the data with the uncertainty of the model due to noise.

3.2 Transitions between Prognostics Types

When using Bayesian transitioning methods there are some assumptions that should be considered. Fundamentally, Bayesian analysis updates a prior belief with new data to get a posterior belief. The general approach to applying a Bayesian method consists of identifying the prior, which comes from the previous prognostics model type. Then observational data is sampled from the newer prognostics type. They are then combined using either the posterior expectation estimates, or linear regression that includes prior information. For most prognostics types involving distributions, the former is used; for anything involving GPM, the latter.

Generally, when solving the posterior estimates, conjugate priors are used to find easy solutions to Bayes' formula based on the distribution of the sampled data. If the prior is not in a conjugate form, it can be parameterized to fit the appropriate conjugate. In essence, this is to assume that the prior can be accurately approximated with the distribution of choice.

The advantages of Bayesian inference are most clearly seen when the uncertainty of sampling data is large compared to the strength of confidence in the prior belief. This also holds true when there is little sampling data available. Eventually in all cases the sampling data should overcome the prior belief. If successfully implemented, the Bayesian transitions should decrease the overall uncertainty, while preserving prior information.

3.2.1 Type I to Type II Transition

For this transition, Bayes formula can be applied using conjugate distributions, **Equation 3-4** and **Equation 3-5**. This is a widely practiced

form of Bayes formula, and is characterized by relatively straightforward equations that are easily referenced. For example, the Gaussian conjugate distribution not only has Gaussian prior and posterior distributions, but also a normally distributed data sample.

Type I RUL estimate distribution can be considered as the prior with the Type II estimate as the sampled data. Considering the fact that most Type I models use a Weibull distribution, a Weibull with known shape parameter with an inverse gamma prior can be used instead. There are also equations that re-parameterize the Weibull parameters, transforming them into the mean and variance parameters of the Gaussian. This re-parameterization can retain much of the original distribution, especially if the distribution is similar to a Gaussian in the first place.

3.2.2 Type I/II to Type III Transition

Bayesian priors can also be incorporated into the OLS model to reduce the uncertainty and increase the stability of RUL estimates. Bayesian statistics combines prior distributions with sampling data to create a posterior distribution. When few data points are available, the model can be easily thrown off and result in wildly varying time of failures. When applied to OLS, the prior parameters from all the failed cases form the prior distribution. The sampled data comes from the censored data.

Mathematically, the transition is achieved by treating a prior RUL distribution as an additional data point.

$$y_* = \begin{bmatrix} y \\ y_{thresh} \end{bmatrix}, \quad X_* = \begin{bmatrix} X \\ MTTF \end{bmatrix}, \quad \Sigma_* = \begin{bmatrix} \Sigma_y & 0 \\ 0 & \Sigma_{RUL} \end{bmatrix} \qquad \textbf{Equation 3-15}$$

For example, if a Type I RUL distribution exists, then y is appended with the degradation threshold and **X** with the mean time to failure based on Type I analysis. The diagonal matrix is appended with a measure of the RUL uncertainty distribution. For both cases, the weight of the prior then depends on two main factors: the variance of the prior against the variance of the data, and the number of samples taken in. If the variance of the prior is small against the noise of the data, the prior b_0 will be weighed more heavily.

3.2.3 Alternatives to GPM via Local Models

As stated in the previous section, one problem in applying the GPM to a failure case is having limited or no previous knowledge of the functional fit of the paths taken by the specific failure mode. For instance, flank wear in cutting tools show an initial steep rise which tapers off to follow a linear correlation between wear and cutting time (Huang 2004). A square root or linear

fit could possibly be applied to great accuracy initially, however, flank wear increases rapidly after passing through the linear region. This means that a square root or linear extrapolation would not be accurately correlated with the future data. For a previously unseen failure mode, such empirical observations of future events would not exist.

Local models can be used to put emphasis on the last known data to try to accurately represent future parameters. This approach, such as the "LEAP-frog" technique (Greitzer, 1999/2001), has been shown to hold some promise in estimating future parameters. However, while the LEAP-frog technique applies local models with extra data from linear independent variables (such as time) and non-linear independent variables (distance driven by tanks), this research combines the local models with other forms of prognostic analysis.

For the PHM data in the previous sections, four additional prediction methods were applied. Two of these methods are the LLR and LWR models explained in section 3.1.5 and the remaining two are those models combined with Bayesian methods similar to method (3). For the Bayes LLR the process is almost identical to method (3) taking into account regression of only the window.

For the Bayes LWR the two weighting functions stemming from the Bayes uncertainty to signal noise ratio and the Gaussian one-sided kernel can be multiplied to form a single variance-covariance matrix Σ. For the Bayes prior "data point" the kernel weighting is given a 1, though this does not have to be the case. It makes sense to give the prior point the standard weighting based on uncertainty as the point does not technically exist as part of the dataset. For the actual data points, because the noise variance is constant, the overall weighting relative to each other is based solely on the kernel weighting. Thus with respect to the current time, X_{j0}, the data set is weighted based on the kernel, and the Bayes prior data point is weighted based on the uncertainties of the models.

Because the data sets are noisy, and many test cases were analyzed, each local model was applied over a range of bandwidths. It is expected that there is a tradeoff between following the ending trends of the data and the uncertainty. For a range of bandwidths from 3 to 50, the lowest average percent errors over all cases were given by bandwidths 3, 4, 6, 8, and 17. While some statistical analysis may be applied to calculate bandwidth in a general or specific case, such analysis is outside the scope of this investigation and will not be considered. Instead, using those five bandwidths, understanding the statistical nature of the problem and that low average percent errors are generally free of outliers which would explode the error values, Figure 3.2-1 compares the different prediction methods: (5) LLR, (6) LLR with Bayes, (7) LWR, (8) LWR with Bayes.

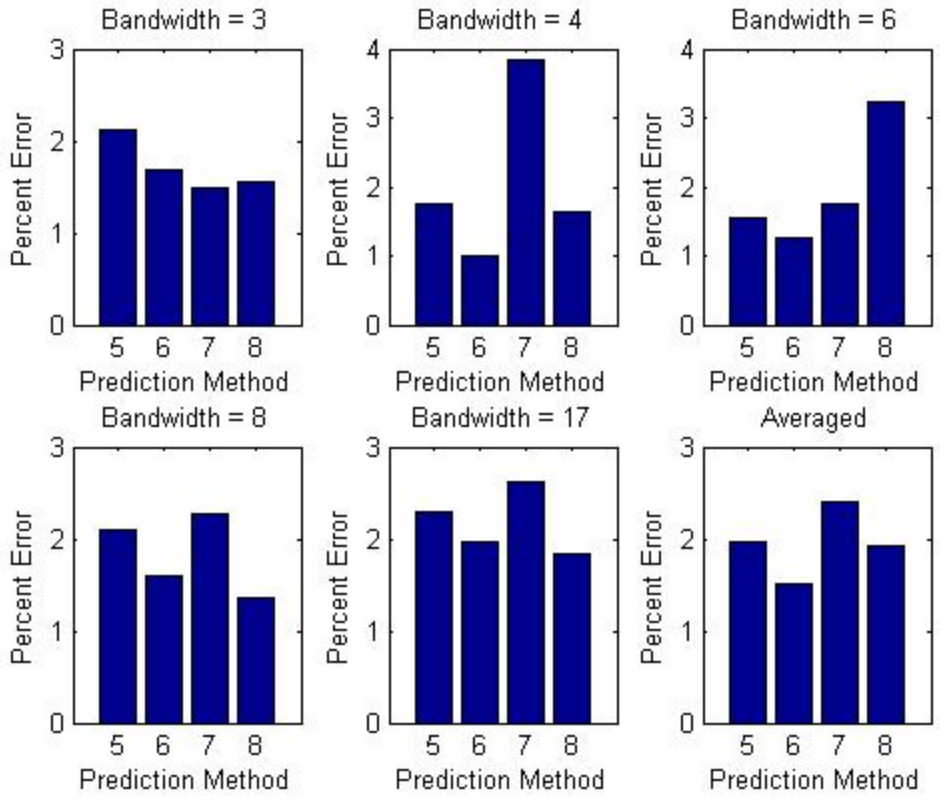


Figure 3.2-1 Comparison of Local Models for Predicting RULs for Varying Bandwidths

It can be seen that, on average, the Bayesian counterparts perform better than the classical local regression models. The LLR seemed to outperform the LWR models. This may be due to heavy biasing of each individual data point at the end in the LWR, especially for the smaller bandwidths, while the LLR balances the noise uncertainty with the need to match ending trends. On average all these models performed less accurately compared to the global GPM. While, for the GPM it was generally known to have a quadratic fit, these local models assume the functional fit is not known. Because the paths are in fact quadratic, there will always be errors when trying to fit linear curves to the data. The more non-linear the data, the poorer the local models should perform, and the more emphasis must be put on the ending data points.

3.2.4 Alternative Methods Conclusions

For this task, two general methods were applied. The first involved a simple Bayesian combination of different RUL distributions. This applies to very general cases in which RUL distributions can be combined to form estimates containing more data.

For the second general method, GPM, 8 different RUL prediction methods were compared on prognostic parameter data. Half were conventional regression methods, (1), (2), (5), (7), while the

other half were original methods involving Bayesian analysis under restrictions of data availability. Method (3) involved the GPM with known functional form and using Type I, or any previous RUL distribution data, as an additional data point in the regression. In method (4) a linear model was applied, to yield a closed-form solution, and the RUL distribution was re-parameterized into regression coefficients. In methods (6) and (8) the same previous RUL distribution was used as an additional data point in regression, and outperformed their non-Bayesian counterparts. The larger errors were due to applying a linear model to a non-linear path, and the fluctuations caused by signal noise.

4. Task 3: Performance Metrics

Performance metrics are methods used to judge how well algorithms work towards some general goal. In terms of prognostics, performance metrics are used to indicate how accurate or robust a particular model is in relative terms of predicting the RUL of a system. The goal of these metrics is to provide some baseline indicator of how well a model performs in either relative or absolute terms such that different models can be quickly compared. Metrics can also be useful indicators of confidence in the output or performance of a model. Without some indication of the overall performance of a model, it is difficult to have confidence in the results. The following section provides a theoretical and practical presentation of selected performance metrics found in the literature as well as some developed within the framework of this research. As shown in this section, for a metric to be useful it must be independent of the model under evaluation and provide an intuitive indication of performance when compared to similar modeling efforts and preferably be ranked on some absolute scale for self-evaluation.

4.1 Performance Metric Demonstration

To apply the metrics we must have data and RUL estimation algorithms. The data that will be applied is shown in Figure 4.1-1, and the RUL algorithms used are Bayesian Linear Regression and Ordinary Least Squares. The RUL algorithm estimates and uncertainties are shown in the right side of Figure 4.1-1; they are RUL estimates of the data. Explanations for the algorithms are available in preceding sections.

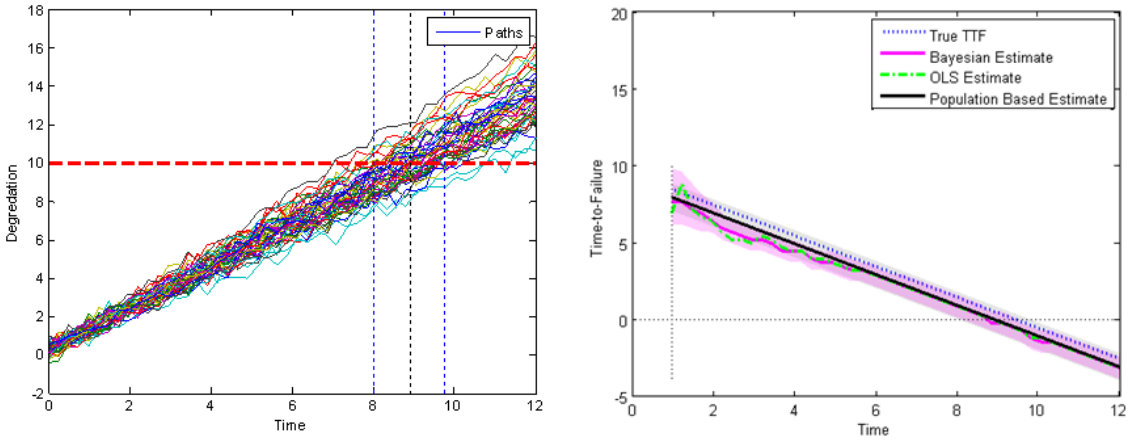


Figure 4.1-1 Data and RUL Predictions and Associating Uncertainties

It is important to note that the methods described are applied off-line and allow for the comparison of various prognostic methods. Application of these metrics requires knowledge of

the true time to failure of the system. These metrics have been coded in MATLAB and are applied to simulated models. In this first part of this section, examples are limited to the linear model in Figure 4.1-1. Simpler metrics, such as Mean Squared Error (MSE), are not addressed since these performance metrics are only related to prognostic methods.

4.1.1 Mean Absolute Percent Error

The first performance metric to be discussed is commonly used and known as the Mean Absolute Percentage Error (MAPE). MAPE is a measure of accuracy through error and is described in **Equation 4-1** [Sexena and Saha, 2009].

$$MAPE = \frac{1}{l}\sum_{i=1}^{l}\left|\frac{\Delta(i)}{R_*(i)}100\right| \qquad \textbf{Equation 4-1}$$

In **Equation 4-1** $\Delta(i)$ is error, $R_*(i)$ is the true RUL value at time (i), and *l* is the range of time that is the difference between the start and end of the RUL prediction. MAPE averages the total absolute percentage error. This metric can be used best under situations where the RUL has a PDF or if there are multiple units under test. Sample estimates for MAPE are available on Table 4-1.

Table 4-1 MAPE Values

Bayes	OLS
14.90	14.90

In this first table based on Figure 4.1-1, the two prediction estimates are very similar to each other so the MAPE value for each is the same. If the two values were different, the lower value would be more desirable as that would imply a lower overall error.

4.1.2 α-λ Performance

The performance metric described simply as α-λ is a method that can be used to determine the quality of a RUL prediction. The α-λ metric measures if a prediction falls within a pre-specified tolerance range of accuracy at time interval λ. This process is defined by:

$$[1-\alpha]*R_*(t) \leq R^l \leq [1+\alpha]*R_*(t) \qquad \textbf{Equation 4-2}$$

In **Equation 4-2**, α is the pre specified performance value, for example α=0.2 would imply a 20% accuracy. $R_*(t)$ is the actual RUL and R^l is the predicted RUL. With this metric, λ

must also be defined as it rescales the time frame, setting the start of the prediction at 0 and the true time of failure at 1. The following equation defines λ.

$$\lambda = \frac{t_\lambda - t_P}{t_{EOL} - t_P}$$ **Equation 4-3**

In **Equation 4-3** t_P is the time at the start of the prediction, t_{EOL} is the time at the end of life, and t_λ is the time at the specified value. The figure below shows three different RUL predictions. The shaded blue area is the specified accuracy zone, α=0.2, in this example.

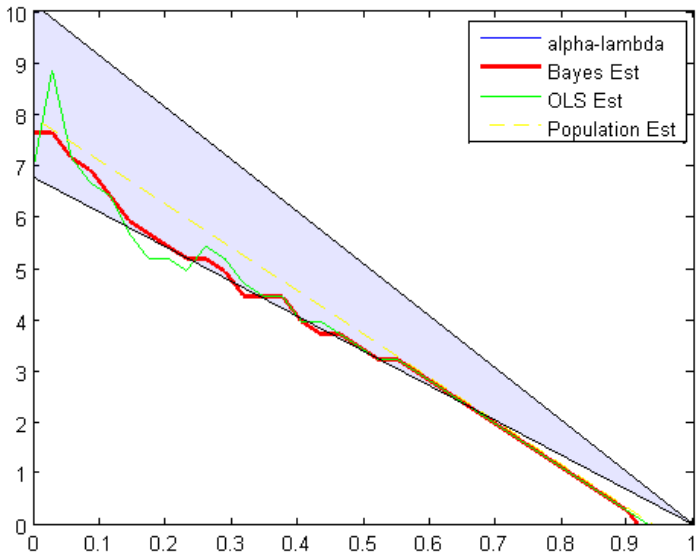


Figure 4.1-2 α-λ Comparison

One item to note in Figure 4.1-2 is that both the Bayesian Estimate and the OLS estimate leave the confidence area (accuracy zone) at a few points. The OLS estimate leaves the blue area first. Both estimates end noticeably outside the confidence region ending about 0.1 before the true time to failure. The advantage of this estimate is how easily it allows the user to compare different RUL estimates and how the progression of the accuracy of the estimate changes over the life of the component.

4.1.3 Prognostic Horizon

The concept of Prognostic Horizon (PH), detailed in Saxena and Saha 2009, is the difference between the time when the prediction starts and ends, given that certain conditions are met. The condition in this case is that the prediction lies within acceptable confidence bounds α as seen in Figure 4.1-2 from the previous section. In other words, the Prognostic Horizon calculates the

difference between the times the RUL prediction first enters the allowable error margin to the time a failure is predicted. **Equation 4-4** defines the Prognostic Horizon. EOP stands for End of Prediction, and "i" refers to the time the prediction enters the confidence region.

$$PH = EOP - i \qquad \textbf{Equation 4-4}$$

The purpose of the Prognostic Horizon is to determine that the prediction is within allowable limits and that the prediction is reliable. The PH can also help distinguish which RUL algorithm is more desirable. The logic is that the larger a PH value is, the more time the prediction has to converge on its answer. Table 4-2 compares Prognostic Horizon values Between Bayesian Linear Regression and Ordinary Least Squares.

Table 4-2 Prognostic Horizon Values

Bayes	OLS
8.449	8.449

In Table 4.2 the two PH values are exactly the same. This will not always be the case, but for the data observed in Figure 4.1 the values are the same. Notice how the OLS prediction initially starts inside the limits but does not stay in. There is debate concerning whether these situations need to be considered when calculating PH. The most important aspect of this metric is that it informs the user of which algorithm has the most time to develop an estimate.

4.1.4 Relative Accuracy (RA) and Cumulative RA (CRA)

Relative Accuracy (RA) and Cumulative Relative Accuracy (CRA) are two concepts similar to the α-λ concept. Relative Accuracy is a metric that determines the accuracy level at time instant λ. **Equation 4-5** defines RA.

$$RA_\lambda = 1 - \frac{|R_*(t_\lambda) - R^i(t_\lambda)|}{R_*(t_\lambda)} \qquad \textbf{Equation 4-5}$$

The same terms that apply in α-λ apply in RA. High accuracy values are desirable with this metric. Values produced are between 0 and 1 with values closer to one being more accurate and lower values being less accurate. Figure 4.1-3 shows a graph of RA estimates as time progresses for two RUL estimate types. The graph shows that early on the Bayesian estimate was more accurate but eventually lost its advantage. The sharp drop in accuracy is telltale of how well the

estimates match the true value at end of life. RA approaches zero near the end of life because the prediction under predicts the actual value.

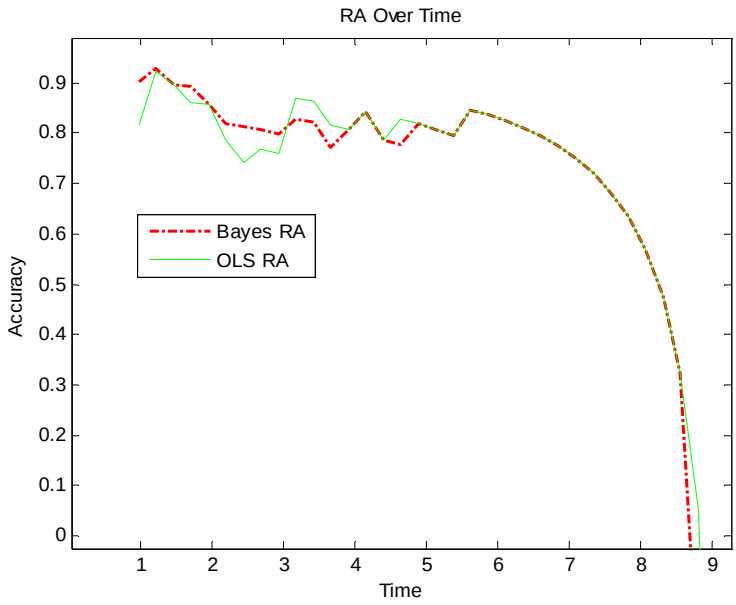


Figure 4.1-3 RA Values from Start of Prediction to End

Cumulative Relative Accuracy is described in Saxena and Saha 2009 as a "normalized weighted sum of relative prediction accuracies at specific time instances." In other words, CRA is the weighted sum of RA values calculated at differing time instances. The formula for CRA is shown in **Equation 4-6**.

$$CRA_\lambda = \frac{1}{l}\sum_{i=1}^{l} w(R^i)RA_\lambda$$ **Equation 4-6**

In **Equation 4-6**, RA_λ is the Relative Accuracy and w is a series of weights. When setting the weights, values further away from the TTF are considered less important than values closer to the end of life. So it is recommended to weight values closer to the end of life prediction higher than values at the beginning of life. The end results are a single value that is representative of the overall accuracy of the RUL estimate. Practical examples are shown in Table 4-3. The RA values are taken at the halfway point (λ=0.5) between the start of the prediction and the end of life.

Table 4-3 RA and CRA

Method	RA(λ=0.5)	CRA

OLS	0.6771	0.213
Bayes	0.6771	0.2155

According to the CRA metric, the OLS method provides a better overall accuracy than the Bayesian method. The cause of this can be seen in α-λ performance. In the figure mentioned, the Bayesian Estimate is generally more accurate early on in life, occasionally dipping below until the two algorithms seem to mirror each other. This mirroring breaks at the end before a failure is predicted. Remember that end of life accuracy values are weighted more heavily than early life values. While the Bayesian method was more accurate than the OLS method earlier in life, those values were weighted less heavily than the later in life accuracy values where the OLS method was better. Because of this the OLS method can be considered more accurate.

4.1.5 Convergence

Convergence generally refers to an estimate's ability to approach an answer. Convergence in the sense presented here, is a performance metric that quantifies how other performance metrics, precision or accuracy based, improve with time. Lower scores mean faster convergence. A weakness of this convergence method is that, when comparing algorithms that start at different times, have differing Prognostic Horizons can skew results. For example, an algorithm that starts later than another and reaches the same estimate could have a lower convergence value, but it does not mean the shorter estimate converged faster. With this method, according to Saxena and Saha, algorithms that start predicting early may seem to have slower convergence then algorithms that start later. The equation that defines convergence is given in **Equation 4-7**.

$$C_m = \frac{\frac{1}{2}\sum_{i=F}^{EOP}(t_{i+1}^2 - t_i^2)M(i)}{\sum_{i=F}^{EOP}(t_{i+1} - t_i)M(i)} \qquad \textbf{Equation 4-7}$$

In **Equation 4-7**, C_m is the convergence value and M(i) is the performance metric value being evaluated. In this (x_c, y_c) is the center of mass of the area under the curve, this makes C_m the Euclidean distance between (t_P,0) and (x_c, y_c) . To summarize these results, the smaller the Euclidean distance is the faster the convergence of the estimate.

In the following applied example, the performance metric Relative Accuracy is being analyzed for convergence. Table 4-4 shows the convergence values for the two methods.

Table 4-4 Convergence Values

Bayes	OLS

1.9496	1.9725

Figure 4.1-4 shows the location of the center of mass for the Bayesian and OLS Relative Accuracy estimates, labeled as CmBayes and CMOLS. Since the Bayesian distance C_m is smaller, the algorithm converges faster making this method arguably better in one area than the OLS method. This result contrasts the findings of the CRA values found in the previous section.

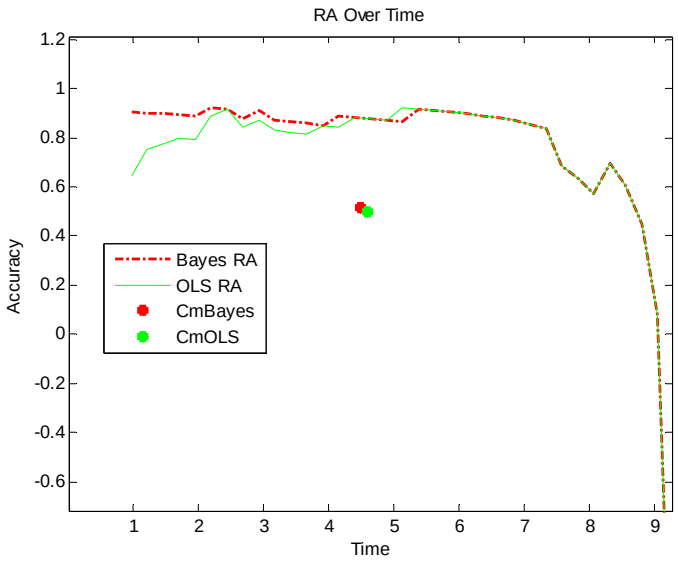


Figure 4.1-4 Centric Values of Performance Metric RA

4.1.6 Uncertainty Spread

Uncertainty spread is a concept similar to that of α-λ. Figure 4.1-5 shows a comparison of the uncertainty spread over time. This graph shows how uncertainty of the predicted estimates changes over time. Initially the Bayesian method is more precise in terms of uncertainty spread. Eventually though, the RUL uncertainty spread evens out as they approach the end of life. The OLS estimate occasionally dips lower than the Bayesian estimate near the end of the prediction, but it always returns to the same value as the Bayesian. This graph can help determine which method has more or less uncertainty.

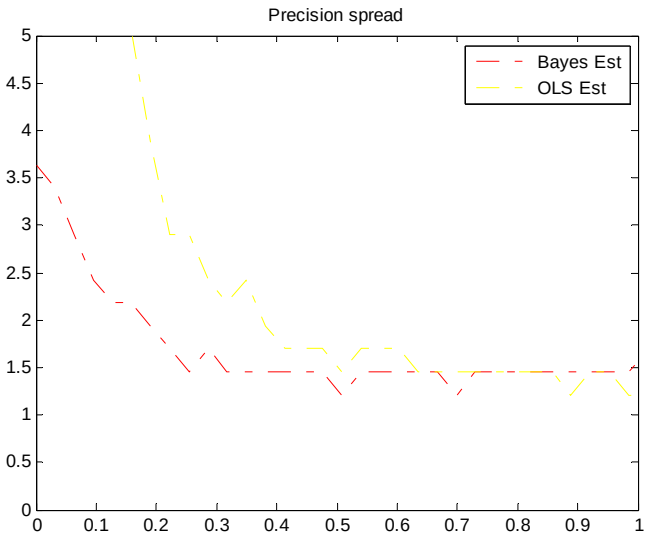


Figure 4.1-5 Spread Over Time

4.1.7 Remarks on Metrics

Setting aside the need for useful and informative performance metrics, there are a variety of real world factors and limitations that need to be considered. Foremost is computational speed. Ideally the software that runs the performance metrics can do so in a timely manner and can keep up with on-line monitoring when these methods are developed. Economic factors of a prognostics operation must also be considered. Ideally the effort to start up and develop prognostics must prove profitable in the short or long term to justify expenditures. Some factors that need to be considered from an economic standpoint are the amount of time required to mitigate or correct a problem detected, the cost of taking such measures, and the cost of failure.

It is not appropriate or necessary to apply all of the performance metrics at once to analyze a model. A systematic approach to analyzing a model must be used and using a specific metric must be influenced by a desired goal. For example, using MAPE and Relative Accuracy to judge a models performance is more redundant than useful.

4.2 Performance Metrics Methodology

A variety of performance metrics were described and detailed using a single example for demonstration. Work in this section will outline and examine the algorithm performance metrics. The algorithm performance metrics consist of four parts: Accuracy, Robustness, Precision, and Convergence. To quickly define each, Accuracy is a measure of error in an algorithm, Robustness is a measure of algorithms resilience to noise or outliers, Precision is a metric that analyzes how close the estimate is to the correct answer, and Convergence is a measure of the rate at which an

estimate improves with time. To develop a proper algorithm performance methodology, the various techniques must be tested and compared.

4.2.1 Performance Metric Hierarchy

To be effective the performance metrics are not to be applied on a case by case basis but by a universally defined method. Doing so creates a consistent standard for which algorithm performance can be judged by. Figure 4.2-1 shows a four-step process for applying performance metrics. The four metrics to be used in order are Prognostic Horizon, α-λ, MAPE, and Convergence. Each metric has a specific purpose. The PH metric produces the time at which an algorithm can yield a result. This is useful when comparing different algorithms, as more time to use data to produce an estimate is preferable. Also, any algorithm used must pass this test to be considered relevant. A negative PH value shows that a prediction starts after the fault occurs, demonstrating that the prediction method is not useful. The α-λ metric is the next estimate test that shows how well the prediction stays within specified confidence bounds. Next the MAPE test serves the purpose of determining how well the algorithm performs quantitatively. Finally the convergence test measures how fast an algorithm converges on its answer. Convergence is the final test because the algorithms predictions must be proven worthy first to understand what the convergence metric means. Bad predictions can converge much faster than good predictions; therefore the quality of the prediction must be ascertained first.

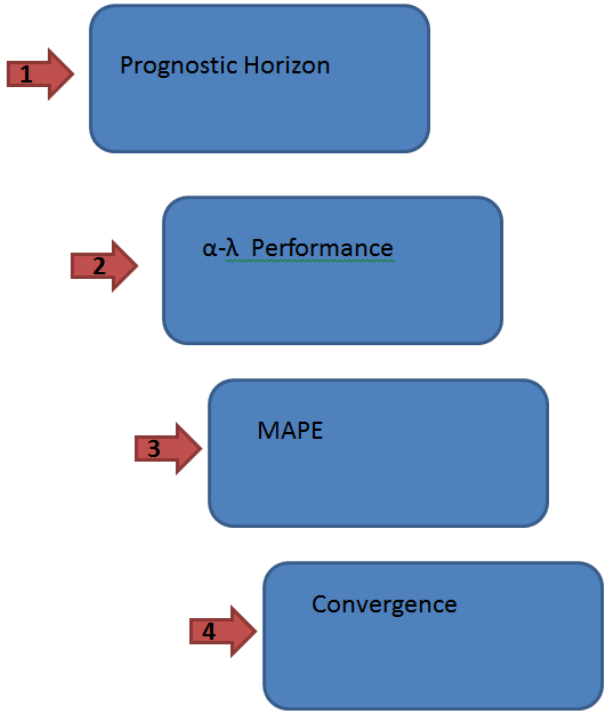


Figure 4.2-1 Performance Metric Hierarchy

This flow chart differs slightly from the hierarchy in the literature. The difference is that the MAPE performance metric is replaced with the Relative Accuracy and Cumulative Relative Accuracy metrics. The main difference between MAPE and RA and CRA is in usability. RA and CRA require user input in selecting the points to compare and in the weighting.

4.2.2 Performance Metric Evaluation

Of the performance metrics demonstrated six will be tested and compared. The five performance metrics are MAPE, Prognostic Horizon, α-λ, Relative Accuracy, Cumulative Relative Accuracy. The metrics being tested are either commonly used or presented in the literature.

To evaluate the performance metrics, simulated data will be produced to create RUL estimates that will be judged by each of the performance metrics. There are three models: linear, quadratic, and exponential. Regression analysis will be used to produce RUL estimates. Each model will have a linear, quadratic, and exponential regression fit applied to it. The purpose of this is to test how the performance metrics handle correct algorithm fits and incorrect algorithm fits. Each model will have one correct fit and two incorrect fits. Ideally, a quadratic model would show that a quadratic algorithm produces the best estimate. The three models are shown in Figure 4.2-2. The linear model has the parameters $\beta 1=N(1,\ 0.02^2)$ and $\beta 0=N(0.3,\ 0.08^2)$. The quadratic and

exponential models have the parameters $\beta 2=N(1.1, 0.1^2)$, $\beta 1=N(1, 0.02^2)$, $\beta 0=N(0.3, 0.08^2)$. Each model has a normally distributed noise with the parameters $\varepsilon=N(0,0.3^2)$.

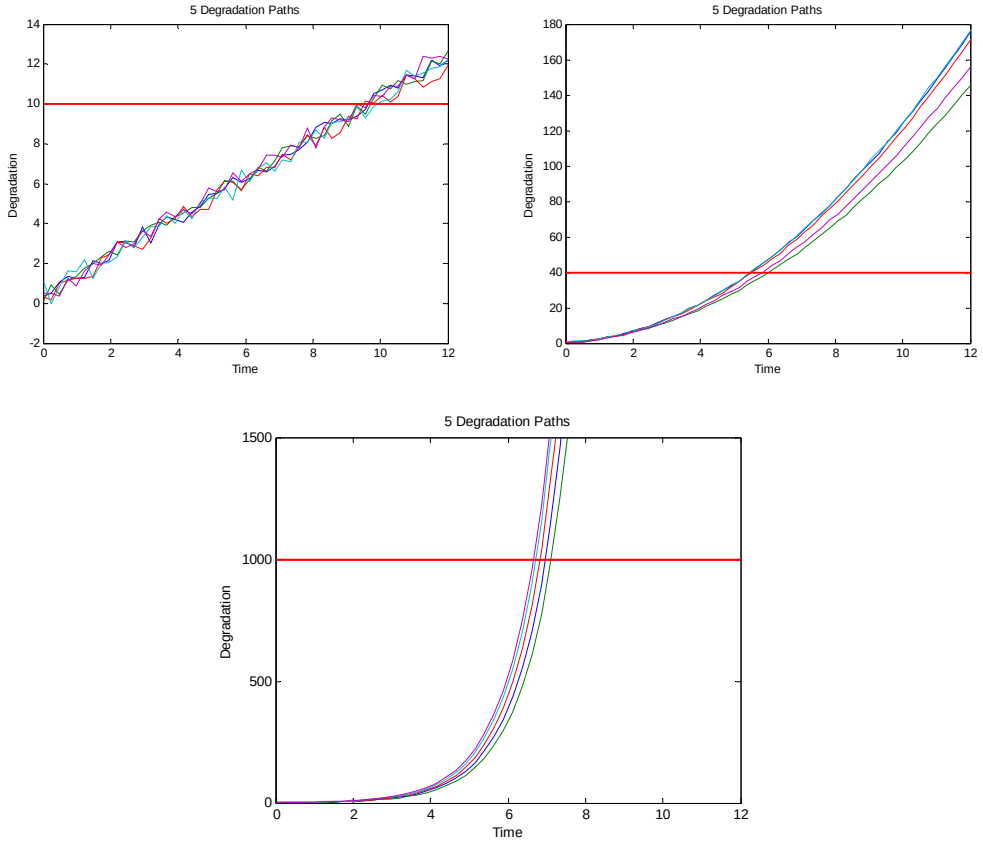


Figure 4.2-2 Linear, Quadratic, Exponential Model

Each model has a separately defined confidence interval. The linear model is set at 10, the quadratic at 40 and the exponential at 1000. The first performance metric to be tested is the α-λ metric. With α value set at 20%, Figure 4.2-3 shows how the estimates perform over time. The three figures in Figure 4.2-3 shows the three estimates, linear, quadratic, exponential, for each of the three models, linear, quadratic, and exponential. In the figure the best fits are shown to be estimates that appear most in the confidence region. According to the three figures the linear model was estimated best with a linear algorithm. The quadratic and exponential estimations were lower than the true value and outside the confidence region. For the quadratic model the linear algorithm generally estimated too high, being far above the true RUL and confidence region. In this case the exponential model proved to be partially reliable as the estimates mostly fell within the confidence region. The quadratic algorithm though, proved to be the best fit as the estimates never left the confidence region. For the exponential model, the exponential algorithm was the only acceptable fit. Both the quadratic and linear model estimates were too high.

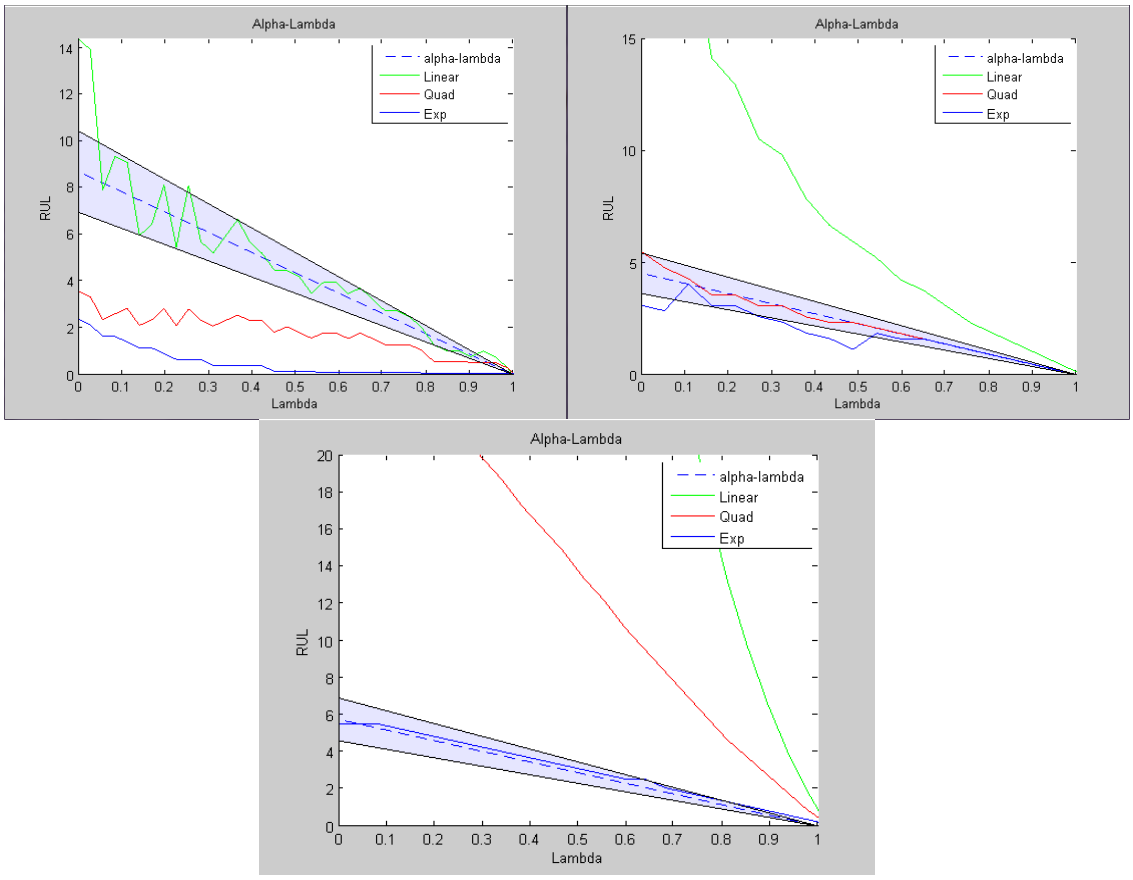


Figure 4.2-3 α-λ; Linear (Top Left), Quadratic (Top Right), Exponential (Bottom)

The rest of the performance metrics are covered in Table 4-5, Table 4-6, and Table 4-7. To quickly breakdown how to read the metrics: with Prognostic Horizon (PH) larger values are better, for Relative Accuracy (RA) and Cumulative Relative Accuracy (CRA) values closest to 1 are best, for Mean Absolute Percent Error (MAPE) lower values are better, and for convergence lower values are best.

Table 4-5 Performance Metric Comparisons: Linear Model

Regression Fit	Linear	Quadratic	Exponential
PH	7.4694	0.6122	-0.1224
RA (λ=.5)	0.9829	0.4229	0.309
CRA	0.249	0.129	0.0279
MAPE	0.2278	0.5702	0.9008
Convergence	3.2644	2.8967	2.6884

Table 4-6 Performance Metric Comparisons: Quadratic Model

Regression Fit	Linear	Quadratic	Exponential
PH	-0.1224	4.5306	4.0408
RA (λ=.5)	-0.5517	0.9957	0.4885
CRA	-0.8215	0.5077	0.4215
MAPE	2.1569	0.0458	0.1595
Convergence	5.1818	2.6305	2.7466

Table 4-7 Performance Metric Comparisons: Exponential Model

Regression Fit	Linear	Quadratic	Exponential
PH	-0.1224	5.7551	5.7551
RA (λ=.5)	-23.623	-2.7445	0.92
CRA	-19.769	-1.2371	0.4114
MAPE	36.7604	3.8102	0.2006
Convergence	31.7781	9.898	3.1382

Each performance metric compared shows that the proper fit generally performs best out of the three fits. The linear simulated data, for example, shows that the linear model has the largest Prognostic Horizon; the best Relative Accuracy at 50% of the components life, the best Cumulative Relative Accuracy, the best MAPE value, but the worst convergence. The convergence metric though is not indicative of a negative overall performance of the linear model when compared to the others. Using this metric in conjunction with the α-λ shows the linear algorithm is the only appropriate algorithm to fit the model and has the only convergence metric that matters.

The quadratic model is unique as it shows that both the exponential and quadratic fits are appropriate. This demonstrates the effectiveness of the techniques as the quadratic prediction is shown to be the better algorithm than the exponential prediction with every metric comparison.

Applying the metric hierarchy to the linear model showcases its usefulness. Starting with the PH value, the linear prediction is the largest (PH=7.4), followed by quadratic (PH=0.6) and the exponential prediction does not pass this test (PH=-0.1). Looking at the α-λ graph for linear

predictions only the linear model is adequately within the confidence region. The quadratic prediction does not pass the confidence test. This point is further illustrated by the MAPE metric, which shows that the linear prediction is the most accurate. The convergence metric shows that the quadratic and exponential algorithms converge faster than the linear prediction but because they do not pass the other tests only the linear algorithm should be considered.

4.2.3 Remarks

The performance metric hierarchy presented in Saxena and Saha, 2009 differs slightly than the process defined in Figure 4.2-1. The change is the reliance on MAPE instead of RA and CRA to determine accuracy values. The current metric hierarchy was chosen for ease of use as the RA methods can involve case-by-case user input.

In testing the methodology, it is shown that the performance metrics can be used to quantify remaining useful life prediction performance. None of the metrics gave contradictory results for improper fits. Close fits, such as quadratic and exponential, were also able to be analyzed allowing for quantitative comparison of effectiveness.

5. Task 4: Integrate PEP with Bayesian Transitioning

Previously developed at the University of Tennessee, the Process and Equipment Prognostic (PEP) toolbox contained tools for rapid creation and implementation of various standard prognostic models. These models utilized the full set of data available within the lifetime of a system, but treated each of the models as disparate entities without any predefined methods for integrating and merging the information of different types of models. As shown in previous sections, this is not the best method in regards to the full utilization of available knowledge of a system. In order to update the PEP toolbox and have it reflect the most up to date algorithms and advances in progressive prognostic lifetime information utilization Bayesian Transitioning has successfully been integrated into the PEP toolbox. This task was divided into two parts. First, the existing PEP functions were enhanced to ease the addition of the new function. Next, the functions were adapted to accept the standard PEP outputs, and accept new optional inputs for the Bayes transition.

5.1 Modifications to PEP

Work to integrate prognostic methods for different prognostic model types into the PEP toolbox and apply Lifecycle Prognostics algorithms with the Bayesian Transitions has progressed in Task 2. A few additional functions have been added to the toolbox and several existing functions have been modified in anticipation of future functions:

- initBayes and fitGPM, both modified to have more flexible inputs.
- runGPM, the uncertainty calculation was updated, and uncertainty output format changed
- runTypeI, one more optional input for reliability percentile was added, equations modified to analytical solutions, instead of numerical approximation
- runprog, runTypeI, runPHM, runMC, and runGPM, all modified to provide give standardized outputs, of both standard deviation, and 95% confidence interval

5.2 Bayes Transitions Functionality

Before the Bayesian transitions could be applied, much of the existing code was modified. In most instances, the outputs of the run functions were standardized to give RUL predictions, standard deviation of the prediction, and the 95% confidence interval in one structure variable as opposed to separate variables. This involved not only changing but also adding to the mathematical models. In a lot of cases, calculations for the standard deviation were added.

The standard deviations were added to make the Bayesian transitions easier, as well as give another method of uncertainty analysis. All Bayesian transitions require a measure of uncertainty. Since the existing run functions do most of the calculations required finding the uncertainty, this was found to be the most efficient method, while also improving the functionality of PEP.

When designing this approach, several factors were taken into consideration. The transitions should naturally fit the current paradigm set by the existing PEP, following the standard use of PEP. Specifically it should naturally take in the models generated by PEP, and accept new data similarly to how the current functions accept data. The user-interface should also be intuitive. The first attempt at creating the Bayesian transitions involved writing a separate function that could take in any two models generated by PEP. The function would analyze which models types were input, and would accordingly apply the correct transition algorithm. However, it became apparent in certain cases, such as when applying the MCMC model, that there may be need for information not stored in the models. This approach posed two problems: the need to add additional information makes the inputs for this function not standard and certain calculations would've been repeated in running both models and the transition algorithm. This violates user-interface simplicity and is computationally inefficient.

5.2.1 Type I to Type II in PEP

The Bayesian transitions were successfully implemented using a modified approach. Instead of separate functions, existing run functions were modified to optionally take in a model of lower type. For example, Type I was not modified in this way, as it is the basic prognostics model. However the MCMC PHM and GPM run functions could take in Type I models, while the GPM could also take in Type II models.

These transitions could be demonstrated again by looking at the 2008 PHM challenge. A Type I model was built and used as the prior for the MCMC and GPM models. It should first be noted that applying MCMC might not be the best approach to this particular dataset. Because the data is unitless, and the background information is unknown, it is impossible to say which, if any, of the inputs would be a good covariate (a representation of the relative operation condition), as such a quantitative analysis of the error has no bearing. Additionally, quantitative analysis was carried out for the Type I to GPM transition presented in the PHM case study. However, the mathematical concepts of both transitions are plainly explained, and verified to yield reasonable results.

In this case, a Type I Gaussian model was initialized using the basic PEP function

```
typeI = initTypeI(TOF,'distribution','normal')
```

where TOF is a vector of the historical time of failures, and the distribution is set to normal, as the default is Weibull. A Weibull distribution could have also been used.

A type II model is then built using the same steps outlined in PEP's previous version, centered on the two PEP functions: MCdata and initMC. Next, RUL estimates using a MCMC model with a Type I transition were made using the same runMC function, but with the additional type I model input.

```
[new_oc map] = MCdata(old_oc);
typeII = initMC(new_oc,'RULcon',0.5);
typeIIout = runMC(typeII,test_oc,typeI);
```

The old_oc is the covariates vector for each case. MCdata maps the covariates into distinct states, new_oc, which is used to build the MCMC model and predicts RUL at the 50th percentile of simulated failures. For comparison, the MCMC was also run without the transition as a control to see how much the Type I prior affected the RUL prediction. In addition to the standard RUL, std, and 95% confidence intervals, runMC returns the posterior variance if Type I is included as an input. The transition was a simple matter of applying the Gaussian conjugate pair **Equation 3-3** and **Equation 3-4**. The Type I provided the prior mean and variance, calculated at that time. In this case time equals number of data points, and weights the MCMC mean and standard deviation.

5.2.2 Transitioning to GPM in PEP

The syntax and approach is similar to that presented in the previous section, in that including the prior involves initializing PEP models. In addition, modifications were made to initGPM, fitGPM, threshGPM, and runGPM to both increase the flexibility of those basic functions, and allow for Bayesian updating when runGPM is called. Some modifications not directly related to the Bayes transitions are listed as follows:

- Calculation and implementation of threshold
- Additional functional fits (cubic and square root)
- Analytical calculations of RUL based on functional fit
- Options for dealing with negative RUL estimates

Additional changes were made as a direct result of the Bayes transitions:

- Posterior uncertainty calculations and outputs

- Priors used as part of the outputs
- Acceptance and syntax of inputs
- Bayesian incorporation of prior models

Applying the Bayesian transitions to the GPM can occur after any applicable Type I or Type II PEP model is initialized. Multiple Type I and Type II models can also be incorporated. In the case of Type II models, additional information about each test case must be included as would be expected when running just the Type II. For instance, for a Type II proportional hazards model, it is expected that the operating condition of each test case is known. These operating conditions must follow immediately after the Type II model to which it refers. The following shows a version of proper syntax when incorporating both a Type I and Type II proportional hazards into a GPM.

```
typeIIIout = runGPM(typeIII, testPar, typeI, typeII, testCondition);
```

Other variations may also be acceptable. As in all the previous cases, initProg and runProg may be used in place of all init- and run- functions, provided that the model type is correctly specified when initializing. In addition, the order in which the Type I and Type II priors are input does not matter as long as they are after the test parameters, and that the Type II information directly follows the Type II model.

As a final note, these Bayesian transitions should not be confused with the already extant path priors. Though both methods are similar in philosophy and implementation, the path priors were already a feature of PEP and use the distributions of path coefficients as priors. These path priors can be used in conjunction with all Bayesian transition methods, and have the combined effect of placing increase importance on the priors, which tend towards "average" predictions.

6. Task 5: Integrate the PEP and PEM Toolboxes

The Process Equipment Monitoring (PEM) toolbox for MATLAB, developed previously at the University of Tennessee, is a package set of tools ideal for building empirical models with the goal of system anomaly detection and quantification. These empirical models emulate a normal system and can be used to predict the expected system behavior for a given set of incoming system inputs and/or sensor indicators. Anomaly detection is largely based on the evaluation of residuals, or the differences between the actual signal and the model output. These residuals can be thought of as quantifications of how much the system deviates from normal behavior, and make ideal inputs for many of the modeling algorithms found in the Process and Equipment Prognostic (PEP) toolbox discussed in the previous section. Developed as separate entities, the integration of these two toolboxes is the logical and natural progression of each as they both are able to provide information necessary in obtaining insight into the RUL of a system. Shown below is a schematic representation of the integrated architecture of both the PEM and the PEP toolboxes.

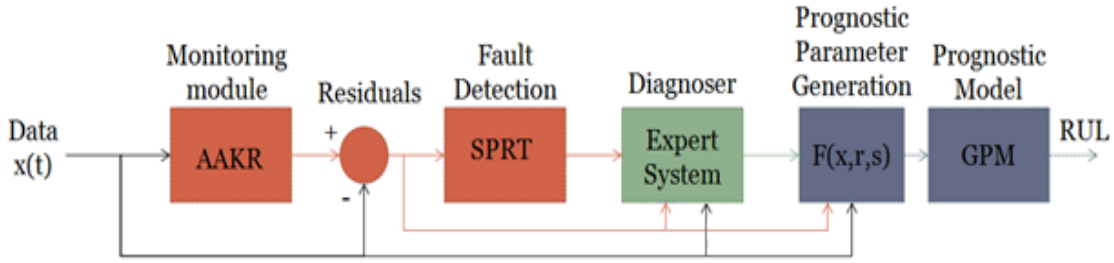


Figure 5.2-1 PEM & PEP Architecture

Those these toolboxes are inherently designed to provide progressive information from one to the other. In regards to the operational lifetime prognostic estimations of a system, the PEM is able to provide the driving indications of the essential Type III effects based models, including when to switch to such a model via fault alarm indicators as degradation increased. This task provides intuitive, user friendly integration of these toolboxes to aid in the rapid development and evaluation of complete reliability and prognostic models.

6.1 Introduction

The purpose of integrating the PEP and PEM toolboxes is to facilitate condition-monitoring tasks while also opening the way to prognostic analysis in a straightforward and intuitive way. There are many benefits to tying together condition monitoring with predictive models. One of the most

important benefits of merging the models together is the ease of data analysis. The ability of the models to predict RUL is directly dependent on the usefullness of the supplied data and any analyses performed. Typically, it is expected that useful information can be extracted from either the residuals or the expected outputs of a condition monitoring system, which can be for prognostic model development.

The Process Equipment Monitoring and Prognostics (PEMP) Suite was developed to aid the user in condition monitoring and RUL estimation. It combines the functionality of both PEM and PEP while providing additional data extraction and interpretation tools. In the MATLAB command line, the PEM and PEP toolboxes can reasonably be applied on the same dataset. However, this is not a true conjoining of the two toolboxes. With additional linking functions as well as a GUI, not only is the user interaction more intuitive, there is less of a burden on the user to understand the entire process.

6.2 The Process Equipment Monitoring and Prognostics Suite

The PEMP Suite is both a scripting based function suite as well as a graphical user interface in MATLAB that allows the user to create empirical models for the purposes of monitoring and prognostics, given the proper data inputs. Using the "Monitoring" set of tools, the user can create different empirical models. Several models can be built and stored to provide easy comparison and can then be used for both fault detection and calculating residuals, which are the deviations between test signals and the model. In the "Prognostics" toolset, various empirical models can be built that are dependent on the type and availability of data. The models are used to calculate the RUL of a system.

This new user interface was the product of merging the algorithms and functionality available in the PEM and the PEP MATLAB toolboxes. These toolboxes provide MATLAB functions to create empirical models, perform fault detection, and calculate RUL predictions. By creating the new interface, many new features were added which make the execution of the tools much more accessible to the user. The graphical capabilities of the interface facilitate automated processes that help the user interpret the modeling process. Relevant figures are automatically generated and the user can control the application of complex algorithms. The following details the major features of the PEMP Suite interface, with further examples and user guides attached to later sections of this document. This is not an exhaustive list of everything that can be accomplished with the PEMP suite but instead highlights the main and expected uses.

6.2.1 Cox Proportional Hazards and General Path Model Transitioning Using Bayes

The PHM is a widely used method in not only equipment prognostics, but in any prognostic field. It is characterized by the use of covariates, values assigned to the system as a representation of the presence of conditional factors pertinent to prognostics. It uses the operating conditions of the system to create updated Remaining Useful Life (RUL) estimates based on statistical analysis and historical failure data.

On the other hand, the GPM is a degradation-based prognostic technique. It combines signal abnormalities into a single prognostics parameter when a fault is present. This degradation parameter is extrapolated to a failure threshold using linear regression, a point at which the system is considered to fail.

A method to combine and transition between the PHM and GPM would not only increase the stability of the RUL estimates of the GPM, but also preserve all prognostic information when multiple sources of failure data are available. Such a method is achieved using Bayesian Ordinary Least Squares (OLS) regression.

Cox PHM:

The Cox PHM is easily referenced, and briefly presented here so that the Bayes transition can be fully documented. It can be summarized by the following equation.

$$H(t,z) = H_0(t)\exp(z * \beta) \qquad \textbf{Equation 6-1}$$

where H is the hazard rate dependent on time and covariate. The H_0 is the baseline hazard rate, and β is a known parameter. The reliability is then given by

$$R = \exp(-H) \qquad \textbf{Equation 6-2}$$

with the conditional reliability

$$R(H|t) = \frac{R(H)}{R(t)} \qquad \textbf{Equation 6-3}$$

This equation gives an RUL estimate based on the current time the system has survived.

6.2.2 GPM

Many methods exist for obtaining the prognostics parameter, referenced here as the generic Y. This usually involves tracking the residuals of the system, observed from initial fault to the end of

life failure. The residuals are the difference between the model and the observed values. The ones pertinent to prognostics can be combined into the single degradation parameter tracked over time. The point at which the system fails can be considered the failure threshold, and is the target for RUL estimations.

The GPM extrapolates to the failure threshold using OLS regression. This takes the form

$$Y|\beta, X, \sigma^2 \sim N(X\beta, \sigma^2 I) \qquad \textbf{Equation 6-4}$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y \qquad \textbf{Equation 6-5}$$

where it is assumed that the predictions of Y are normally distributed about Xβ and equal variance.

6.2.3 OLS with Bayes

The path parameters can also be assumed to have a normal distribution.

$$\beta|\sigma^2, y \sim N(\beta^*, V\sigma^2) \qquad \textbf{Equation 6-6}$$

This equation reflects the reality that each individual system follows its own degradation path, these are then modified to allow for varying uncertainty.

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y \qquad \textbf{Equation 6-7}$$

$$V\sigma^2 = (X^T \Sigma^{-1} X)^{-1} \qquad \textbf{Equation 6-8}$$

To include prior estimates of the parameters, Y, X, and Σ can be appended accordingly.

$$Y = \begin{bmatrix} y_i \\ \beta_0 \end{bmatrix}, X = \begin{bmatrix} x \\ I \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_y & 0 \\ 0 & \Sigma_\beta \end{bmatrix} \qquad \textbf{Equation 6-9}$$

The priors are again assumed to follow a normal distribution with mean β_0 and variances. These variances are represented in Σ_b as a diagonal matrix. The I_k matrix is the identity with the length equal to the number of coefficient parameters.

6.2.3.1 Transition between the PHM and GPM

To transition between the PHM and GPM it is assumed that the RUL distribution of the PHM is fairly normal. This is to facilitate the mathematics, and is a reasonable enough assumption as a prior. The RUL distribution from the PHM can be found first using the failure distribution.

$$F = 1 - R$$ **Equation 6-10**

Because the data is discrete, the probability mass function f is found for F. This probability mass function's weighted mean and weighted variance characterize the RUL distribution.

Using this distribution, the prior equation can be substituted with

$$Y = \begin{bmatrix} y \\ thresh \end{bmatrix}, X = \begin{bmatrix} X \\ RUL_{PHM} \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_y & 0 \\ 0 & \Sigma PHM \end{bmatrix}$$ **Equation 6-11**

This substitution assumes a normally distributed point at the PHM estimated RUL. The variance is expressed in Σ_{PHM}.

This Bayesian transition can be used in conjunction with the previously presented Bayes OLS using path parameter priors. By appending all known priors, any source of information available can be used to calculate the posterior RUL estimate. As with all Bayesian statistics, the stronger the prior measured by the tightness of the prior distribution, the more influence it has over the posterior. Care must always be given in the appropriate prior weight according to the analysis. If the individual is more important, then the GPM should be weighted heavier. As more data is input, the priors are naturally decreased in influence. However, when little GPM data is available, the more statistically based PHM will be weighted heavier. This Bayes approach provides a natural transition from the beginning of fault to end of fault.

These Bayesian transitioning methods are currently being tested on data taken from test beds. These include the pump and heat exchanger accelerated degradation experiments. In addition, while the newly developed PHM transition has been codified into the PEP toolbox, it has undergone more robust testing and refinement.

7. Task 6: Performance Metrics Development

Recent effort has been focused on the standardization of prognostic model performance evaluation based on meaningful criteria that can be used to compare the output of prognostic models not only within given application, but across the field of predictive engineering [Saxena 2008]. Unfortunately, despite this large step forward in the evaluation of prognostic models, there has been a huge oversight in the fundamental evaluation criteria. Until now, many of these tailored prognostic model parameters focused almost exclusively on the evaluation of individual cases output by the model and implicitly reported this as a metric for the overall acceptance criteria for that model. This work seeks to correct this oversight, and presents variants on several well-known performance metrics that represent, and are in fact built upon, a multitude of known cases to which the prognostic model has been applied.

Specifically, five separate metrics have been defined to sufficiently characterize the output predictions of a prognostic model: Mean Absolute Error (MAE), Weighted Error Bias (WEB), Weighted Prediction Spread (WPS), Confidence Interval Coverage (CIC), and the Confidence Convergence Horizon (CCH). Each one, detailed below, captures a key aspect and desirable quality of prognostic predictions that can be quickly, easily, and intuitively compared amongst separately developed models in order to rank and rate their output performance. These metrics are built upon the errors and uncertainty associated with each prediction set, rewarding the minimization of both.

7.1 Mean Absolute Error

The Mean Absolute Error (MAE) is by far the easiest metric to compute, and in many ways is the most intuitive to understand. Unfortunately, this metric could also be argued to be the least informative about the overall performance of the model. Defined earlier, MAE is the average absolute difference between the model prediction P_i and the true Remaining Useful Life (RUL_i) at all times t and for all historic query cases i.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{t=1}^{T_i}\left|P_i(t) - RUL_i(t)\right| \qquad \textbf{Equation 7-1}$$

The advantage in the use of this metric is the MAE is determined in real times units that may be used to compare actual system lifetime. Similarly, one could also calculate the standard deviation of the prediction absolute error for a measure of the spread of these errors.

These metrics are useful for comparing separate models built upon similar data, or data from systems with comparable lifetime scales, but give no clear indication of prediction performance without some context to the data. Another shortcoming of these metrics is that they have the implicit underlying assumption that the errors are normally distributed, which may not always hold true and in some rare cases lead to very misleading indications. These standard formula metrics are also inflexible to individual requirements about the specifications of the predictions, and can be largely susceptible to outliers.

Using MAE in the evaluation of prognostic prediction performance is severely limited because even though this metric provides a meaningful way to evaluate the expected error in the lifetime of the system it falls short when the system reaches end of life. The RUL estimates are more important near end of life than at the beginning and the MAE doesn't take this into consideration since it is just a mean of the RUL estimates. The remaining metrics introduced and described in this work help to overcome and fill in the gaps left by MAE and similar standard metrics.

7.2 Weighted Error Bias

The Weighted Error Bias (WEB) is the first of the lifetime percentage based metrics. WEB, as defined in Equation 7-2, is a measure indicating the effective bias in all predictions as a percentage of the total unit lifetime.

$$WEB = \frac{100}{N}\sum_{i=1}^{N}\sum_{t_i(1)}^{t_i(n)}\frac{\left(\hat{P}_i(t) - RUL_i(t)\right)}{TrueLifeTime_i} \qquad \textbf{Equation 7-2}$$

From this equation it becomes evident that WEB is very similar to MAE except in two important respects. First, it is tallied and reported as a percentage of the total lifetime of the individual query unit, i. This allows for the intuitive inspection of the performance of a series of predictions without the need for some contextual setting. A model whose predictions yield a 10% WEB would be expected to be better than one with a 25% WEB regardless of the units or time scale involved. This also has the added benefit of implicitly scaling the errors, such that similar deviations from the true Remaining Useful Life (RUL) values for short-lived components would be weighed heavier than those in longer-lived units, even within the same historic data set. This is important, as an error of 20 time cycles is less significant if the unit in question lives 300 cycles as opposed to if it only lives 100 cycles.

The second difference is in the explicit importance weighting, w_i, of the different errors based on their time in the lifecycle of the historic unit. This importance weighting can easily be tailored to the specific needs or desires of the end user, but a clear emphasis on the end of lifetime is the most meaningful for prognostic predictions. A 10% error near the beginning of unit life when there is 85% of life remaining gives plenty of time to act and take corrective actions, where a 10% error with only 5% of life remaining could result in an unexpected failure if the unit were expected to live through the remaining cycles. An example of a weighting function that accurately reflects this end of life importance is the Gaussian Kernel Function with a mean value set to the lifetime of the unit and a bandwidth set to 50% of that lifetime.

Clearly the optimal value for this metric is zero, indicating that the average prediction value is centered on the true RUL. In fact, this metric can also provide a crude method for model improvement by simply subtracting the indicated percent bias from all the model predictions. This method is extremely crude, and in nearly every case it would be better to readjust actual model parameter in order to create better predictions, but in the absence of that option, this can help to improve estimations. A less crude version of this would be to map and subtract the WEB at various points during the lifetime of the historic units. Binning error values based on their associated percent of unit life can help with this.

7.3 Percent Error Value Binning

The final three prognostic prediction performance metrics rely on quantifying the estimated uncertainty of prognostic predictions throughout the total lifetime of a query unit. In order to do this effectively, the 95% confidence interval (or some similar level of confidence interval) needs to be calculated at various points throughout the unit lifetime. One of the more straightforward methods for doing this is to create a set of bins evenly divided between 0 and 100% of system lifetime, and place each calculated percent error in the bin corresponding to the true percent of unit life for that error. In other words, first calculate the percent error for a given historic prediction, $P_i(t)$, such that the percent error is the difference between the predicted RUL and the actual RUL divided by the query unit ,i's, total lifetime.

$$\%Er = 100 * \frac{\hat{P}_i(t) - RUL_i(t)}{TotalLyfTime_i} \qquad \textbf{Equation 7-3}$$

Next note the corresponding percentage of actual lifetime (POL), defined by the current time, t, divided by the current unit's total lifetime. Finally place the calculated percent error into the POL bin whose edges, B, are defined as:

$$B_{LOWER} < POL_i(t) < B_{UPPER} \qquad \textbf{Equation 7-4}$$

Repeat for all historic predictions across all query cases, placing them in to the same series of corresponding bins. By virtue of the numbers all being converted into percentages, allows for the direct comparison and inclusion of these similarly located values with proper importance weightings applied based on their lifetime.

Once this series of regular serial bins is populated, a 95% confidence interval around the mean value can be calculated from the 2.5% and 97.5% percentiles of the error set for each bin. Clearly, these percentages can be altered to suit specific application requirements if necessary. Additionally, the expected value for each individual bin can be calculated, creating an expected error bias that maps throughout the lifetime of a unit. As mentioned before, this bias map could be used as a rough means of improving prediction performance in the absence of better or more sophisticated methods.

7.4 Weighted Prediction Spread

It is an accepted truth that the quality of any calculated prediction can be defined by its associated uncertainty. Thus it follows that the quality of any prediction model should also be defined by its associated uncertainty. Additionally, much like the model prediction error and bias, not all points during the lifetime of the query system should necessarily be treated with equal importance. The predictions of Remaining Useful Life (RUL) made by a model are considerably more important near the end of the system's life than they are at the beginning of life, since near the beginning of life there is comparatively much more time to react and compensate, or mitigate any impending faults or failure inferred from the prognostic model.

The spread of model predictions at various points in life are an important factor in the total considerations of the uncertainty of a series of prediction. The prediction spread for each binned point of system life is calculated as the difference between the upper and lower bounds of the corresponding 95% confidence intervals from the binned error values discussed previously. Using the exact same importance weighting function as the Weighted Error Bias (WEB), the Weighted Prediction Spread (WPS) can be defined as:

$$WPS = 100 * \frac{\sum_{bi=1}^{\#Bins} W_{bi} * CI_{bi}}{\sum_{bi=1}^{\#Bins} W_{bi}} \qquad \textbf{Equation 7-5}$$

In this equation, the weighting function is based on the center value for each reference bin, such that each bin importance weighting, W_{bi}, is defined by the Gaussian kernel.

$$W_{bi} = \exp\left(-\left(\frac{Bin_{bi} - 100\%}{50\%}\right)^2\right) \qquad \textbf{Equation 7-6}$$

Notice that the typical normalization factor associated with Gaussian kernels is rendered unnecessary due to the inherent normalization factor included in the definition of WUS. Although a kernel bandwidth of 50% is shown, other bandwidths can easily be substituted to accommodate specific needs. All the factors and values associated in the metrics based on the binned interval error values are listed and manipulated as percentages, allowing for quick intuitive evaluation of the effective important uncertainty of any given prediction set.

With this metric, a 0% WPS alone would seem to indicate absolute certainty in all predicted values, but this may be misleading. In fact, all this indicates is that all predictions made are exactly the same, based exclusively on the percent RUL of the system in question. Because of this fact, this metric should always be coupled with the WEB to indicate if the predictions do, in fact, have enough spread to cover the true RUL (i.e. WPS $\geq$ WEB). A more explicit and useful metric evaluating this coverage is the Confidence Interval Coverage (CIC), which should also be calculated, and is discussed in the next section.

7.5 Confidence Interval Coverage

Another important indication of the quality of a prediction set generated by any model,is whether or not the confidence interval of the prediction spread covers the true RUL. This effectively incorporates information relating to both the error bias and the error variance at given points in life. The metric is simply defined by the total percentage of binned error sets whose 95% confidence interval contains the true RUL. This is more rigorously defined:

$$CIC = 100 * \frac{\sum_{bi=1}^{\#Bins} RUL_{true} \subset B_{bi}}{\#Bins} \qquad \textbf{Equation 7-7}$$

This equation is interpreted as the sum number of true percent RUL values that are contained within their corresponding error bin set, divided by the total number of bins, and multiplied by 100 to convert to a percentage. This additional metric verifies the total accuracy of the prediction set. An optimal coverage of 100% shows that the true value of any prediction is contained within the prediction spread or approximate confidence interval of the prognostic model's predictions. When coupled with the previously detailed metrics, this gives a solid expectation of the accuracy and expected effective error over the total of system life predictions. The final vital element not conveyed by these metrics is the explicit end of life accuracy and precision. The Confidence Convergence Horizon fills this void.

7.6 Confidence Convergence Horizon

This final metric captures and quantifies the end of life quality of both the precision and accuracy of a prediction set. A 10% Confidence Convergence Horizon (CCH) identifies the percentage of system RUL beyond which, all prediction confidence intervals are both less than 10% of the total system life and contain the true RUL. In other words, the CCH identifies a predicted time to failure where, once reached, it and all additional predictions can be trusted to be within 10% of the true value. Obviously a CCH of 100% would be optimal, showing that all predictions within the query set are within less than 10% of the true values.

Although this seems to be a rather stringent criterion to meet, it is nonetheless very important. This horizon is a quick and intuitive identifier of the region of most confidence for a particular prediction set. Unfortunately, like any single descriptive metric, the CCH has the potential to be misleading if it is not considered along with the other metrics defined in this section. As an example, consider a model that predicts the RUL of a system within >10% during most of the system life, but due to an unlikely artifact, exhibits an 11% bias at the end of life. This model would produce a CCH of 0%, as there is no point in time when all following predictions can be trusted to be less than 10%. This does not, however, mean that the model produces unusable or even inaccurate results.

Each of the listed metrics contains and expresses vital information required to develop a full understanding of a models performance, but it is often convenient to assign a single quantitative value of "goodness" to a particular model and prediction set. Described in the following section is a method for developing such a unifying metric.

7.7 Total Score Metric

There has been proposed a sort of hierarchical ranking of some of the previously developed metrics [Saxena 2009]. To some degree, this work is able to eliminate the explicit need for this hierarchical system, and in its place supplies a single aggregate scoring metric to rank the overall performance of a particular prognostic model's output predictions. Of the metrics detailed in this paper, four in particular can be merged to give a singular quantitative value of "goodness" for a prognostic model prediction set. These metrics, Weighted Error Bias (WEB), Weighted Prediction Spread (WPS), Confidence Interval Coverage (CIC), and the Confidence Convergence Horizon (CCH), detail a particular yet vital aspect of the total historic prediction produced by a given model. With this in mind, and given that each one of these metrics has been constructed to be listed in similar units of percent Remaining Useful Life (%RUL), it becomes obvious that a basic composite of these metrics can yield a meaningful, simple, and direct measure of the quality of a model prediction set. These metrics can easily be applied for quick quantitative comparison of multiple models' prediction sets.

$$TotalScore = N * \begin{bmatrix} 100 - WEB \\ 100 - WPS \\ CIC \\ CCH \end{bmatrix} \qquad \textbf{Equation 7-8}$$

Note that in this equation, both the absolute value of the WEB and the WPS are subtracted from 100 to reflect that the minimums of these values are the desired quantities. N is any normalized vector weighting the importance of the four metrics. For simplicity and intuitive interpretation of the resulting number, a vector of [.25 .25 .25 .25] results in a simple mean of the metrics, which can then be used to rank the model's performance out of 100%. Some of the model metrics contain similar information. This is not useless redundancy, but instead reflects the increased importance of these aspects when the metrics are combined. For example, if a set of model predictions exhibit 0% CIC, that prediction set would also by definition exhibit a 0% CCH. Coverage of the correct RUL within a confidence interval is one of the most important criteria any prognostic model should meet, so with the standard weighting set, the best total score the model could produce would be less than 50%, reflecting that the model has never produced a correct answer.

7.8 Prediction Metric Example Cases Studies

To help further clarify and explain the prediction metrics, consider a standard pump and motor system with a mean failure time of about 275 operating hours with two common modes of failure with different mean failure times. Three separate simulated models were built to predict the RUL of these motors. The first is based strictly on statistical conditional time based probability of failure. The second two are built to simulate more effects based modeling types. In order to compare the three models, each one simulates a set of 100 predictions about similar sets of query cases and has the metrics detailed above applied to those prediction sets.

Shown in Figure 7.8-1, the Model 1 prediction set for all 100 cases completely overlay one another. This is expected and due to the fact that this model's output is based exclusively on the current lifetime of the queried system.

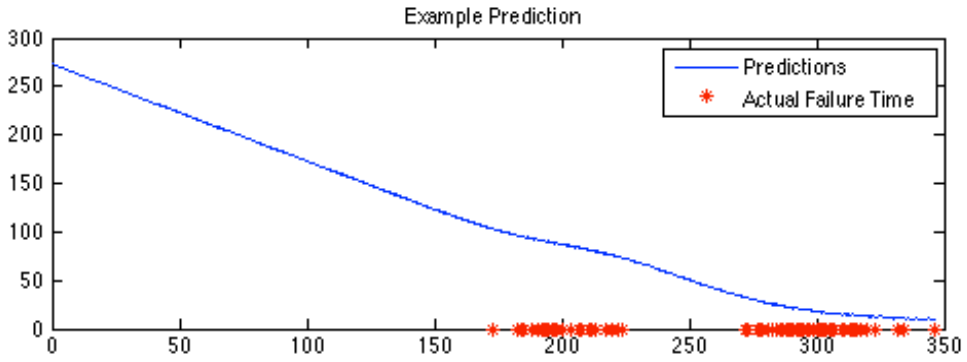


Figure 7.8-1 - Model 1 RUL Predictions

Despite the fact that each of the predictions for each of the individual cases is exactly the same, they represent varying percentages based on the true queried system's lifetime. This is accounted for in the calculations of the performance metrics shown in Figure 7.8-2.

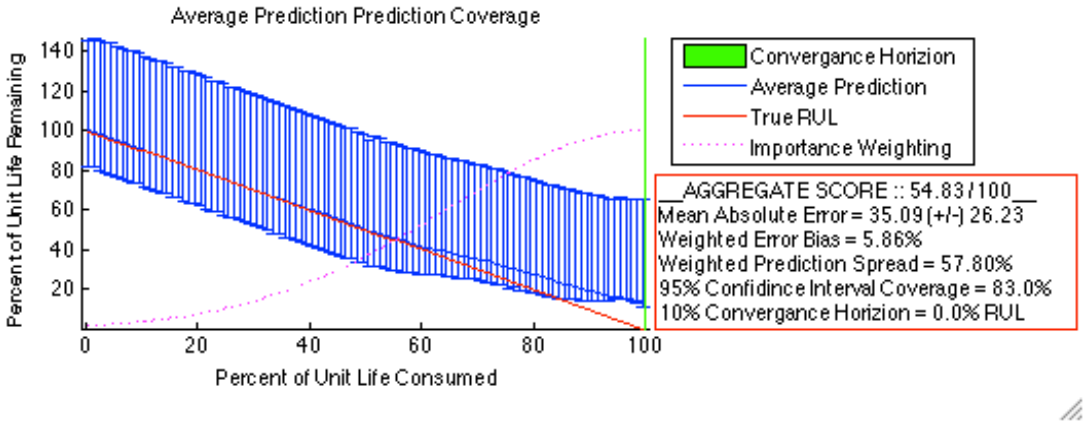


Figure 7.8-2 - Model 1 Prediction Performance Metrics

The most intuitive and easily understood metric on this figure is the Mean Absolute Error (MAE), listed as 35.09 hours with an associated standard deviation of 26.23 hours. The MAE gives a good basic understanding of how much error to expect out of the model, and is good for comparing models that are run against the same data set. However, the three example models presented here are run with differing query data sets. The sets are taken from similar sets of pump systems, but the individual units and their true total lifetimes are different. Although MAE could be used to compare these models and prediction sets since the time units and expected average lifetimes are the same, the percentage-based metrics are more appropriate and generally informative.

The most prominent prediction evaluation tool in this figure is the binned error average estimate and their associated 95% confidence intervals represented by the blue error bars. This contains the most total and useful information about the prediction set. From this chart it is obvious that early in life the model predicts the correct percentage of RUL on average, but also has high uncertainty, meaning it may in fact never predict the exact true RUL for a particular unit. This inference is confirmed by examination of the end of life binned error as the average model prediction value departs from the true RUL line at around 62% of life consumed (38% RUL) and loses even the 95% prediction interval coverage at around 85% of life consumed (15% RUL). Because this is a strictly time based model, it helps to confirm that the model is unable to precisely predict individual systems' RUL, instead only calculating the average RUL over all historic systems. Although this chart of binned error is useful and contains a wealth of information, it does require some degree of examination and analysis in order to compare different model sets. The other percentage based prediction metrics provide that analysis.

The effective bias for this model, as calculated by the Weighted Error Bias (WEB) from Equation 7-2 is 5.86%. Again, this can be seen in the binned error analysis as the average estimation line begins to deviate from the true RUL line, particularly near the end of life. For this system, that means that there is an effective average bias of about 16 hours, but this does not mean that the expected error is 16 hours. This value, as well as the Weighted Prediction Spread (WPS), is considered an effective value because of their applied weighting function shown in the previous figure as a magenta dotted line, which allows them to be more effective at ranking the predictions. If for some reason, the more literal average values are needed, the same equations and metrics can be applied with a simple adjustment of the weighting function. This prediction set's WPS is listed as 58.07% of life, reflecting the fact that there is a considerable amount of uncertainty associated with the predictions.

The final two metrics listed are the Confidence Interval Coverage (CIC) and the Convergence Horizon (CH). Reported at 83% and 0% respectively, these indicate that although the model uncertainty covers the true RUL 83% of the time, it never continuously falls within 10% of that true value towards the end of the unit's life.

All these metrics can be combined in order to give this model's prediction set a total ranking of 54.83% out of a possible total score of 100%. This should not be read as an indication that the model's total accuracy is around 50% or that only 50% of the model's estimations are trust worthy. Instead, this metric shows a quantitative evaluation of the model's performance for this prediction set. It is a quick and relatable evaluation of the model's "goodness" which can easily be used to compare against other models or other prediction sets. For example, if Model 1 is compared to Model 2 shown in Figure 7.8-3, one can quickly see that Model 2 has a total performance score of 75.02%, which is much better than Model 1's 54.83%.

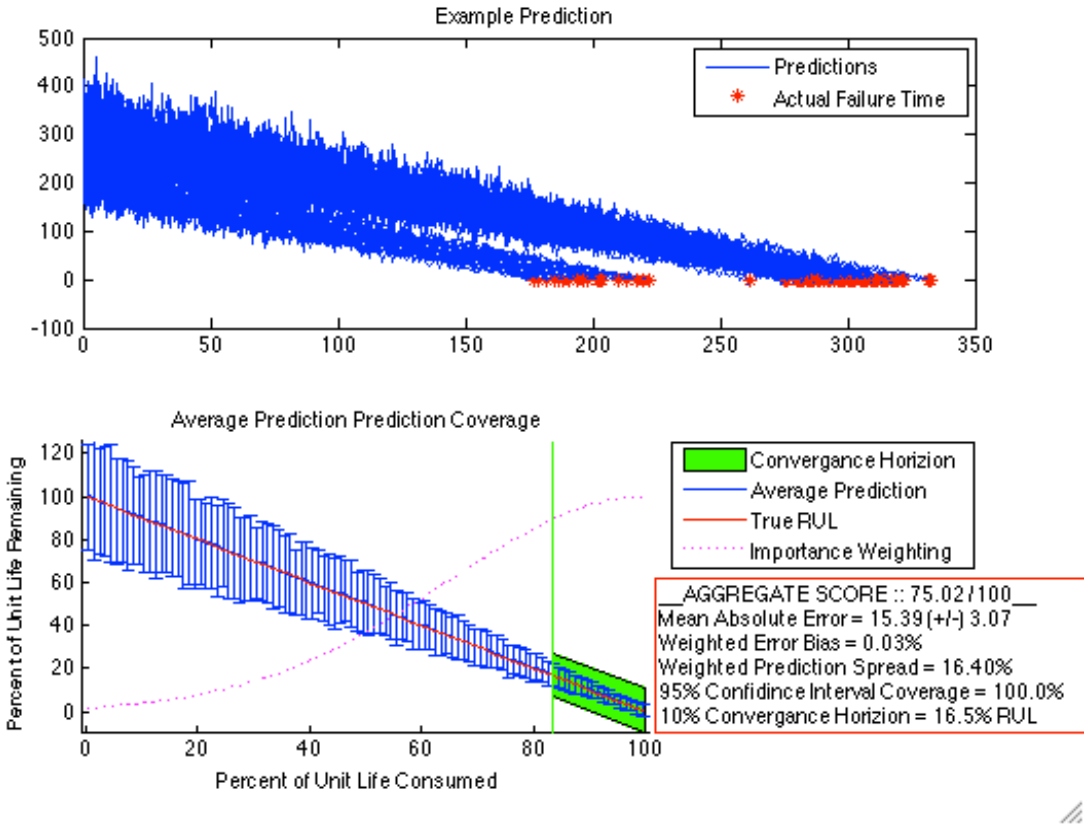


Figure 7.8-3 - Model 2 Predictions and Metrics Evaluation

Looking at the individual metrics, it becomes clear why this model is ranked better. Firstly, it has 100% CIC with a 16.5% CH meaning that not only is the model more accurate overall, but it also shows that the accuracy improves to near the end of life. Next, the effective prediction spread is 16.4% of life, much lower than Model 1's WPS. Finally, Model 2 has virtually 0% effective bias, meaning that all the predictions are centered on the true RUL.

These metrics give a quick, effective, and qualitative method for comparing two different models, and if that were the only end goal the analysis could stop there. However, if there is opportunity to change and improve the models, which created the prediction sets, then the scalar metrics alone may not give the complete picture. Consider the prediction set developed by Model 3 in Figure 7.8-4.

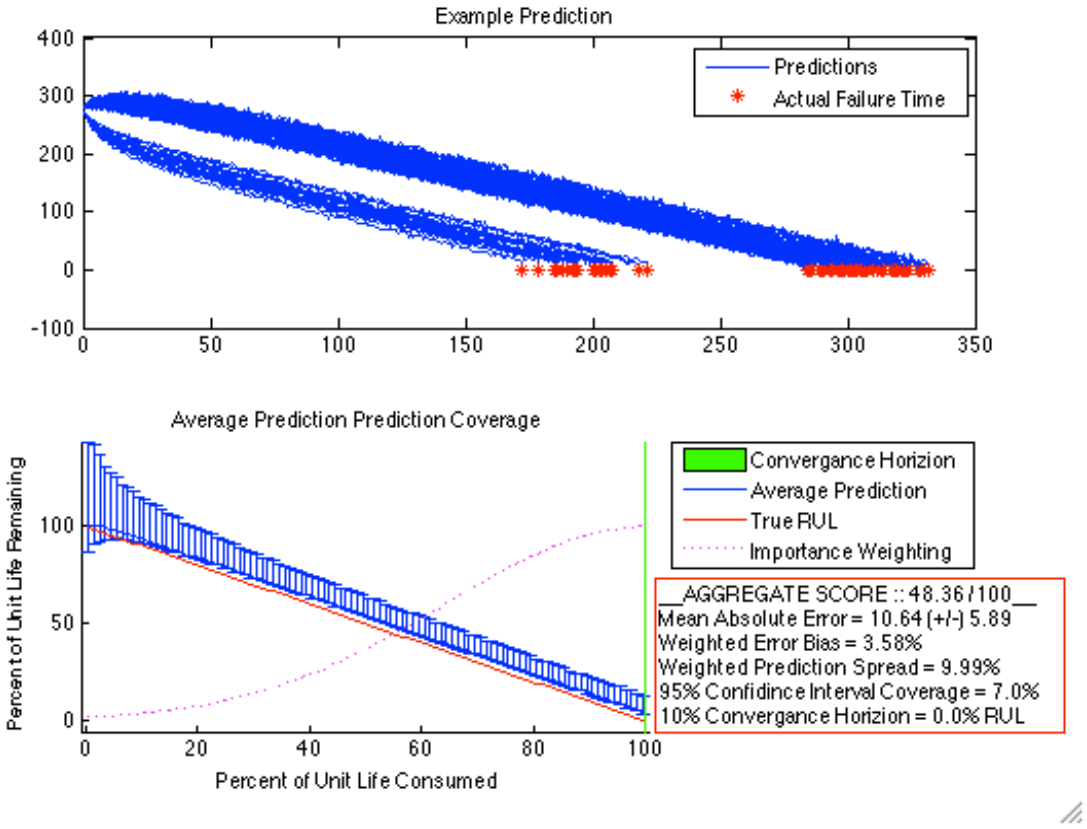


Figure 7.8-4 - Model 3 Predictions and Metrics Evaluation

Model 3 has a total performance score of 48.36%, indicating that is worse than either of the two previous models. In fact, the only metrics which it outperforms both the other models are MAE and the WPS. Unfortunately, these alone would not necessarily merit further investigations into the development of this model. However, when the total binned prediction value map is investigated, it becomes clear that by removing a small bias in this model, these predictions would be expected to outperform either of the previous models. This same conclusion could be inferred from the scalar metrics, but a graphical examination of the binned values map is both more expedient and informative.

7.9 Summary and Conclusions

In general, estimation uncertainty comes from two sources, the variance and the bias. Unfortunately, when evaluating the performance of prognostic estimates, uncertainty alone is not enough to fully characterize the performance of the model estimations. Some temporal emphasis based on the current and expected lifetime of the unit or system in question must be taken into account. Shown here is a standardized method and metrics for evaluating prognostic estimations and incorporating scaled temporal importance of the measures of uncertainty reflecting both

accuracy and precision. The scalar metrics presented in this work help to provide clear and concise comparisons between both similar and dissimilar prognostic model prediction sets. In order to demonstrate and help to visualize the underlying meanings of each of the metrics, three separate sets of predictions made from three separate simulated prognostic models. From the results listed in Table 7-1 it is clear that Model 2 is the best performing model by a large margin.

Table 7-1 – Summary of Model Comparison Results

	Total Score	MAE	WEB	WPS	CIC	CH
Model 1	54.83%	35.09 Hrs	5.06%	57.80%	83.0%	0%
Model 2	**75.02%**	15.39 Hrs	0.03%	16.40%	100%	16.5%
Model 3	46.36%	10.64 Hrs	3.58%	9.99%	7.0%	0%

Further, Model 3 shows great potential for improvement via a simple bias removal as can be inferred from the low Weighted Prediction Spread (WPS) coupled with the results of the binned prediction value map. A quick summary of each metric is listed below in Table 7-2.

Table 7-2 - Metrics Summary

Metric Name	Quality Aspect Reflected	Units
Mean Absolute Error (MAE)	Accuracy	Real Time Units
Weighted Error Bias (WEB)	Timely Precision	Percent of Unit Life
Weighted Prediction Spread (WPS)	Timely Accuracy	Percent of Unit Life
Confidence Interval Coverage (CIC)	Accuracy	Percent of Unit Life
Confidence Convergence Horizon	Timely Accuracy & Precision	Percent of Unit Remaining Useful Life

(CCH)		
Binned Prediction Value Map	Timely Accuracy & Precision	Percent of Unit Life

These metrics can quickly be adopted into a standard prognostics evaluation toolbox. A MATLAB based function for evaluating and displaying these metrics has already been developed and has undergone rigorous testing and evaluation.

8. Task 7 and Task 8: Experimental Setups

In order to further develop and evaluate many of the methods, tools, and algorithms investigated in this project, a collection of physical real-system data was required. This data needed to be analogous to physical systems present in industry operated plants with similar collected signals, and expected fault modes. This section presents the development of experimental test-beds for the validation of Prognostic techniques for the RUL analysis and prediction. The updated Process and Equipment Monitoring (PEM) and Process Equipment Prognostics (PEP) toolboxes, developed at The University of Tennessee, will be applied to the data collected from these experimental setups. Both anomaly detection algorithms and lifetime prognostic models complete with data driven transitions are utilized to predict the different PROaCT laboratory systems' Remaining Useful Life (RUL This section presents the development and data collection process of these test bed facilities over the course of this project.

8.1 Electric Motor Aging Experiment

Motor Electric Power Research Institute (EPRI) originally funded the purchase and testing of ten 5hp (horsepower) electric motors. These U5P1G U.S. Electrical Motors/Emerson general-purpose industrial motors were chosen as low cost analogs to the high power induction motors found throughout industry. Table 8-1 lists the full nameplate specifications of the motors used.

Table 8-1 Motor Nameplate Specifications

Brand	U.S. Motors/Emerson Premium Efficient General Purpose Industrial Motor
Model Number	S5P1A
HP	5
RPM	3600
Volts	208-230/460
"C" Dim	15.9
Frame	184T
Full Load Amps	13.6-12.1/6.1
Full Load Efficiency	88.5

Apx. Wt. (lb)	70
SF	1.15
Phase	Three
Motor Mount	Foot Mounted
Enclosure	TEFC
Type	General Purpose Industrial Motor

Each of these 3-phase, 3600 RPM motors were subjected to a cyclic thermal aging process, designed to induce accelerated insulation breakdown and corrosion within the motors. These fault modes have been selected as the most prominent and costly for inductions motors. The original accelerated aging plan was adapted from previous work performed by Upadhyaya (1997) and IEEE Standard 117, "IEEE Standard Test Procedure for Evaluation of Systems of Insulating Materials for Random-Wound AC Electric Machinery Degradation and Testing Plan."

According to IEEE Standard 117, several testing procedures may be chosen to perform accelerated degradation testing of motors. For Class F insulation (the type of insulation in the motors that will be tested during this project), the recommended testing time is 32 days at 170 degrees C. For this testing, the motors have been divided into two groups, one which will be heated to 160° C, and one at a lower temperature of 140° C to create multiple condition sets. The temperature set lower than the IEEE standard for the "hot group" provided a slower, more realistic evolution of any degradation mechanisms and related features of the motors and allows for more accurate tracking and estimation of the degradation curve of the testing motors.

8.1.1 Experimental Methodology

Previous work concerning a similar aging program of electrical motors lead by Dr. Upadhyaya was also referenced during the design of this experiment. In that work, a condensation chamber was not used for moisture accumulation as recommended in IEEE Std 117 due to the difficult task of creating multiple condensation chambers for the motors being tested at the same time. The motors were instead quenched in a tub after being allowed to cool for six hours after the thermal degradation testing. After the quenching, the motors were allowed to dry overnight before beginning thermal degradation testing again. Since the primary focus of the experiment was on thermal degradation, this amount of moisture accumulation was sufficient for testing purposes, even though the motors did not use the procedure recommended by IEEE Std 117. Each accelerated aging cycle has been designed to take just over one week, and performed in one of 3 identical EW-52402-91 Lab Companion Economy Mechanical Convection Oven. A detailed listing of the thermal aging cycle process is listed below.

Thermal Aging

1. Heat motor in laboratory-grade oven for 72 hours
2. Remove and allow to air cool for 6 hours.
3. Quench in enclosed shallow water pool for 15 minutes
   (this acts as the suggested replacement for the recommended humidity chamber)
4. Immediately place back in the oven and heat again for 72 hours.
5. Air cool for 18 hours before performing "Startup Testing"

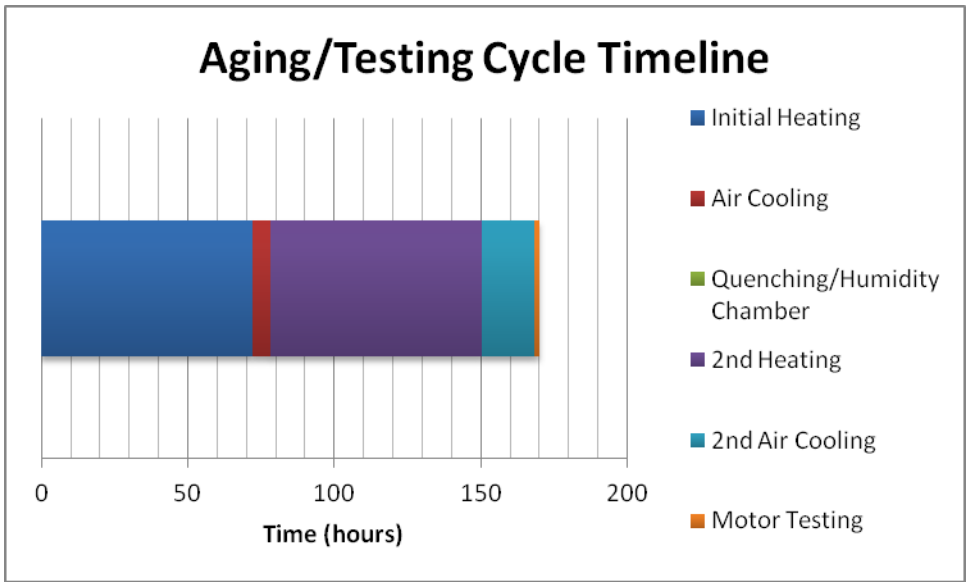


Figure 8.1-1 Time Requirements for Aging and Data Acquisition

Figure 8.1-1 details the time requirements for both an individual thermal aging and data collection cycle in hours. This chart shows that each individual aging and data collection cycle will take just under a week of total time.

After undergoing each thermal aging cycle, the motor is mounted on a test bed, connected through an elastomeric coupling to a Winco generator, and instrumented with a data collection system to collect various key signals from the motor during both the transient startup and periodically during the steady state operations. The full list and specification of sensors is shown below:

- Compact DAQ: NI cDAQ 9178 (8 slots)
  - 3x phases of current
    - Fluke i200s Current Clamps: Input/Output 600 V CAT
    - NI DAQ Module 9234
  - 3x phases of voltage

        - NI DAQ Module 9225
    - 2x accelerometers (90 degrees apart: vertical and horizontal from ground)
        - 0.0002 g resolution, +/- 50 g measurement range
        - NI DAQ Module 9234
    - Load current and voltage for the dynamometer
        - NI DAQ Module 9225
    - Motor speed from tachometer
        - ICP Laser Tachometer: Reads up to 30,000 RPMs
        - NI DAQ Module 9205
    - Thermocouple located in motor near the stator winding.
        - Omega K type 30 gauge Chromel Alumel surface mount thermocouple
        - Typical use range: 95C – 1260C
    - Acoustic sensor
        - PCB 130D20 ICP array microphone
        - Sensitivity 45 mV/Pa
        - Frequency Response (+/- 1 dB) 100 Hz to 4 kHz

The current and voltage from the connected output generator are also monitored, but are largely ignored for the work in this paper. A photo of the setup is show in Figure 8.1-2.

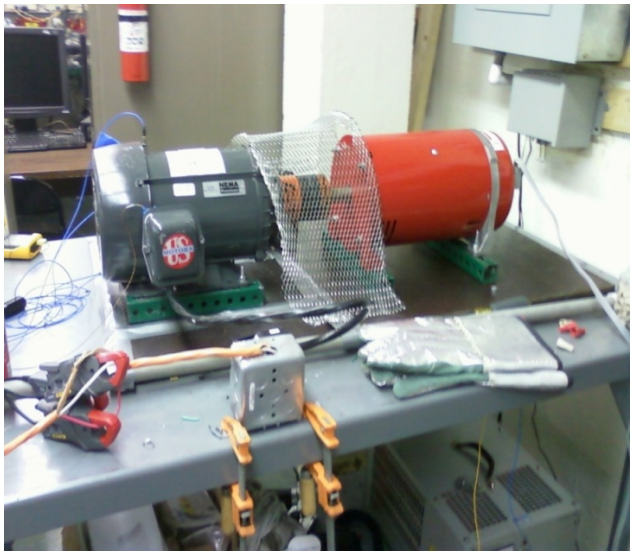


Figure 8.1-2 Motor Testing Setup

Figure 8.1-3 shows the LabVIEW created user interface for the data collection.

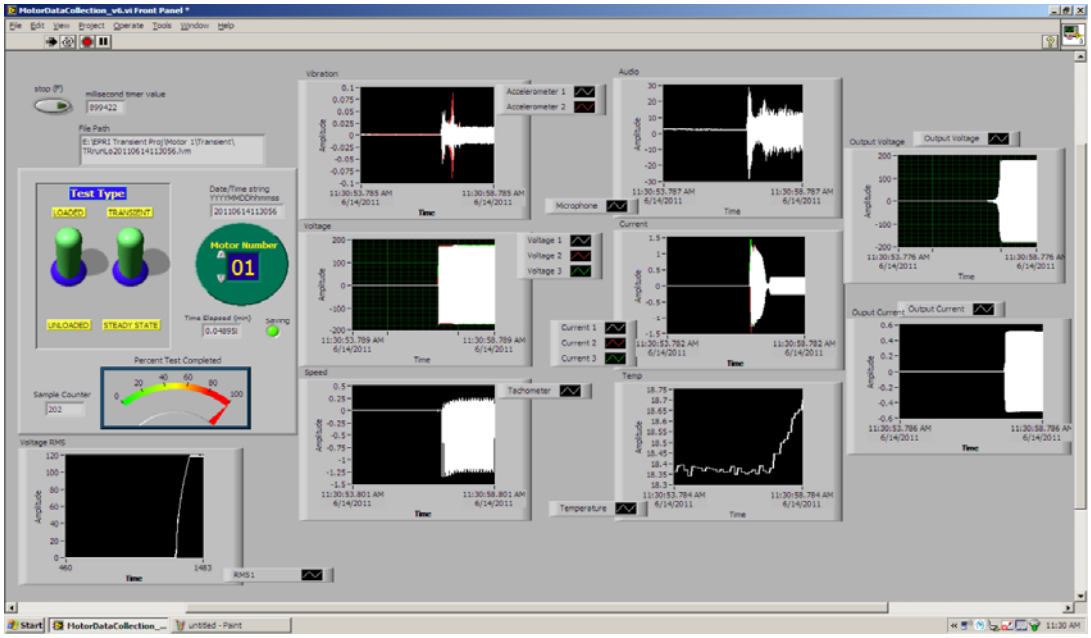


Figure 8.1-3 LabVIEW Data Collection Interface

The end sink for the supplied energy is a resistive load bank connected to the generator. Multiple startups are collected between aging cycles in order to help smooth trends and reduce measurement error in the analysis. The data is collected at just over 10kHz for slightly less than two seconds at a time with a NI LabView interface. The LabView data acquisition software collects and manages all signals as double precision floating point values before the collected data is stored in a comma delimited text file for flexibility in current and future data analysis.

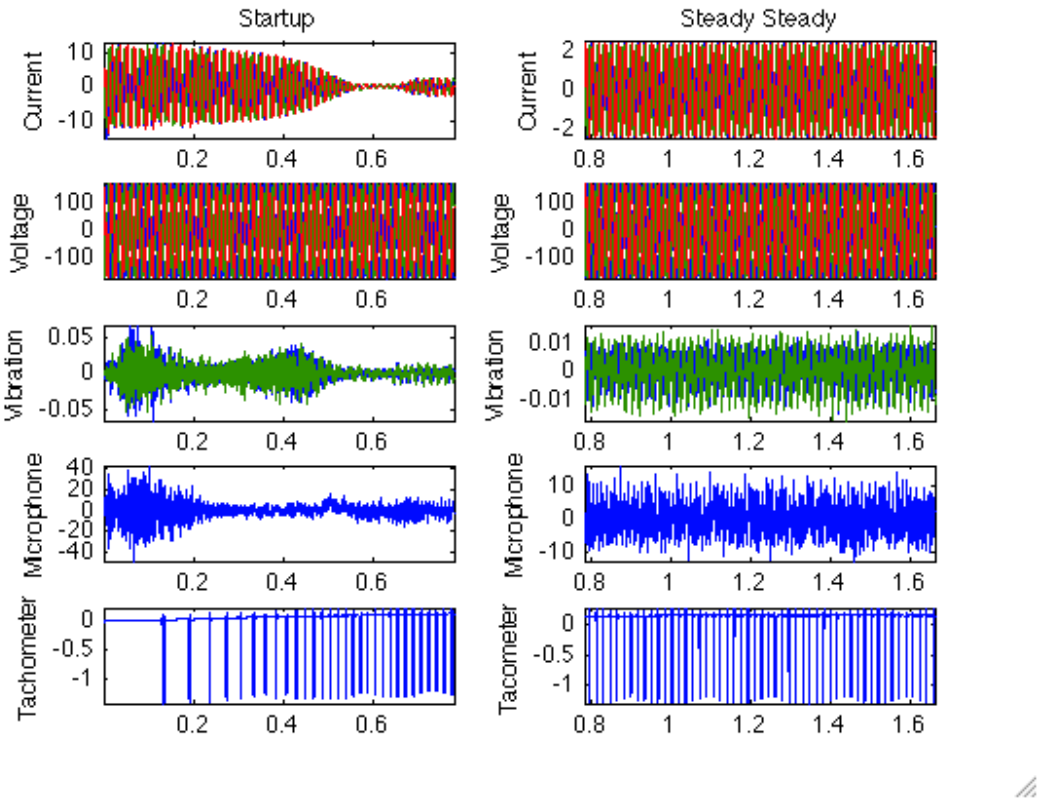


Figure 8.1-4 Exemplar Collected Signals at Beginning of Life

An example of the collected data, both startup and steady state conditions is shown in Figure 8.1-4. Shown in this figure, there are significant differences between startup and steady state. Nearly all of the signals undergo significantly higher magnitudes during startup than those experienced at steady state. This additional stress forms the basis for the hypothesis of this work that transient conditions can exacerbate fault indications and provide additional useful prognostic information regarding the system.

8.1.2 Initial Fault Mode Forensics

After completing the testing from the initial set of motors several important aspects of the testing procedure became evident. First is that the main mode of failure involved the degradation of the shaft bearings. Secondary modes were largely trivial and centered on maintenance and setup induced faults.

Following failure, motors were disassembled, inspected, and the results documented. The motors were disassembled by removing the end-bells and rotor, and then extracting the shaft and opposite end bearings with bearing pullers to prevent possible additional damage to the bearings.

The primary mode of failure was excessive starting torque cause by bearing degradation. Thermal damage to bearing grease and seal material during both accelerated aging and testing resulted in grease failure, namely separation of lubricant from stabilizers and introduction of material from bearing seals into the bearing cavity. Inspection of bearing internals revealed that virtually no lubricant remained in any of the bearings. In the case of the catastrophically failed bearing, which resulted in complete fragmentation of the inward-facing shield and seal, bearing internals were essentially dry. This bearing retained all balls, but much of the ball cage was ejected. All motors showed deposition of lubricant stabilizers below the bearing housings on both end-bells. Additionally, all bearings exhibited seal failure, with seal material found hardened inside and outside the bearing.

It was also noted that all motors exhibited external wiring degradation where insulation had failed or wiring was exposed for connection to the test stand. Failure of terminal wiring insulation and breaks of terminal wiring generally occurred during handling, and sometimes due to interphase arcing. In most cases, this was repairable, though in multiple instances arcing generated sufficient energy to destroy all ex-casing wiring, precluding repair.

These observations from the initial motor aging experiment prompted several changes for the aging procedure of the motors tested during this project. Additionally, a larger emphasis was placed on generating internal stator insulation failures. The changes implemented are summarized in the sections below.

8.1.3 Motor Aging Experiment Test Procedure Modifications

The originally tested motors failed predominantly due to lubricant failure-induced bearing degradation and electrical failures associated with degradation of the stator wiring harness. Neither of these failures modes were the main focus of this experiment, therefore to mitigate the processes that led to these modes, additional testing procedures as well as alterations to the testing conditions were implemented for the second set of motors. These alterations were aimed to prevent the mechanical degradation of the connection wires, provide an expected level of maintenance to the installed bearings, and accelerate the insulation aging of the test motors.

The first of the new procedures removed the necessity to manipulate the stator wiring harness in the course of testing, thus eliminating the mechanical degradation of the external wire connectors. The primary leads are each wired into permanent ceramic connection housings with a set of simple replicable leads that can then be connected to the main power source. The ceramic terminal block is external to the terminal box, protecting the stator wiring. These leads have

soldered terminal leads to reduce corrosion of the wiring, and were arranged such that further manipulation of the internal wiring was unnecessary, thus preventing that mechanism of degradation.

The second of the new procedures concerned the bearing failures in the previous motor aging test. The lubricant problem was mitigated by utilizing motors that feature re-greasable bearings, which is an uncommon feature for the small motors under test. A survey was conducted of industrial literature regarding greasing, but it was found that for greasing applications in extreme environments, the grease schedule is highly unpredictable. For an accurate determination of the appropriate grease volume, specific process knowledge in the environment for a specific bearing must be gained. For this experiment, such information is unavailable, likely because the rated temperatures of typical greases for bearing applications are below the temperature of environmental exposure. Also, unlike typical systems, the experimental motors have had their cooling capacities disabled. This was done primarily as a safety measure, as the vendor only provides plastic cooling fans that can disintegrate violently during use after a single heating cycle. The greasing schedule is expected to prevent bearing failures during the test, in spite of the service conditions. Multiple motors have required clean-outs of the shaft end grease lines, but all have shortly been returned to service with noticeable reductions in vibration.
A final alteration to the testing procedure was to increase the aging temperature of the motors to 180 degrees Celsius and eventually to 200 degrees Celsius. This increase was aimed at accelerating the internal insulation degradation. In conjunction with the bearing maintenance it was thought this would allow the motor stator degradation to become the dominant failure mode in the aging motors.

8.1.4 Phase II Motor Selection

The primary criteria for selection of a motor for Phase II testing is a motor that is small enough to make testing economical and proceed in a timely manner. Secondly, the motor needed to permit the grease to be maintained in a condition which prevents bearing failures. Very few small induction motors permit re-greasing, as small motor service conditions combined with the excellent quality of modern commodity bearings dictates that the economical solution for most industrial consumers is a sealed bearing. The Baldor EM3613T is a three-phase induction motor utilizing the same bearing specification, the same frame, and similar electrical specifications. The bearings are re-greasable through standard Zerc fittings. Regreasing will be performed prior to each test. Grease contamination was minimized by cleaning the Zerc fitting and adjacent motor

surfaces prior to greasing, purging the grease gun prior to greasing, and greasing in accordance with vendor procedures to prevent over or under-greasing, both of which can be deleterious to bearing service life.

Table 8-2 Baldor EM3613T Induction Motor Specifications

SPEC. NUMBER:	36G271S042G2
CATALOG NUMBER:	EM3613T
FL AMPS:	11.8/5.9
208V AMPS:	--
BEARING-DRIVE-END:	6206
BEARING-OPP-DRIVE-END:	6205
DESIGN CODE:	A
DOE-CODE:	010A
FL EFFICIENCY:	88.5
ENCLOSURE:	TEFC
FRAME:	184T
HERTZ:	60
INSULATION-CLASS:	F
KVA-CODE:	L
SPEED [rpm]:	3450
OUTPUT [hp]:	5
PHASE:	3
POWER-FACTOR:	91
RATING:	40C AMB-CONT
SERIAL-NUMBER:	--
SERVICE FACTOR:	1.15
VOLTAGE:	230/460

8.1.5 Motor Aging Experiment Results

Unfortunately each of the motors in this secondary set failed due to undesirable mechanisms before producing any detectable fault indicators. These failures largely centered on the eventual degradation and failure of the external lead wires and the associated housing terminal box. Post mortem forensics showed nearly no visibly notable internal insulation degradation. Of the ten motors tested, none showed conclusive signs of internal degradation, instead all but on suffered eventual failure due to lead wire failure. The remaining motor, Motor 12, failed due to a ground fault during an improperly setup test procedure. Each of the implemented new test procedures to prevent bearing failure and mechanical degradation of the external lead wires appeared to be effective. However, though this extended the lifetime of these motors significantly, it did not

ultimately preclude the wiring lead box from being the primary cause of failure throughout the testing. A summary of the total testing cycles for each of these motors is provided in Table 8-3 below.

Table 8-3 - Motor Aging Cycle Summary

			Cycles At Temperature:		
Motor	Start Date	End Date	180oF	190oF	200oF
M11	1/11/2013	3/17/2014	38	8	1
M12	2/12/2013	3/14/2014	30	10	2
M13	2/12/2013	3/5/2013	3	-	-
M14	2/11/2013	3/18/2014	30	10	1
M15	2/12/2013	3/14/2014	30	11	2
M16	2/26/2013	3/21/2014	30	11	1
M17	3/1/2013	3/25/2014	23	10	3
M18	3/4/2013	3/25/2014	30	7	3
M19	3/4/2013	1/20/2014	33	3	-
M20	3/4/2013	1/27/2014	26	5	-

The inferred reason for this is that the bulk of the induced thermal aging is not being directly deposited in the internal windings of the motors. Both the external housing and the exposed wires suffer the largest thermal gradients during testing due to their direct contact with the atmosphere. While this is unavoidable with the aging procedure detailed above, it did prompt second large revision to the aging regime. Detailed in the section below, the final procedure mechanically overloads the motors during testing to incite huge electrical current draws and thus increased internal temperatures directly at the internal windings.

8.1.6 Motor Aging Mechanical Overload Testing

In response to the continued unavoidable undesirable failures due to housing shorts of the motors under the thermal aging regiment, a new aging procedure was enacted. By mechanically overloading the motors during operation and allowing them to overheat due to the increased electric current draw, the internal wiring of the motors will bear the bulk of the induced stress prompting desired electrical failures. By focusing both electrical and thermal stress directly within the stator in this manner will also minimize the induced stress to the bearings and outer housing lead wiring, both of which have proved to be the dominate mechanisms of failure under the previous thermal aging cycles.

Two motors were acquired and stressed through mechanical overloading. The connected output generator was set to demand 7 horsepower during each test. This is 40% in excess to the maximum recommended load for the tested motors. In order to ascertain the most effective method for mechanically overstressing the motors, the two motors were run in two different operating modes. The first was run with intermittent stops and starts; one per every hour of operation. Not only is transient signal information collected during these times, which can add the potential for an additional level of data analysis in future extensions of this work, but this also adds an additional mechanism for stressing the motor. The second motor was run continuously at overload operations until failure.

The first of these motors, designated as Motor 21, began to show signs of degradation within the first 40 minutes, as visible smoke began to emit from the wire terminal box. This is assumed to be the early indications of internal insulation vaporization. The overall temperature of the motor increased steadily throughout the aging test, which ended at 2 hours and 17 minutes from the initial motor energizing. The motor underwent a total a three startup-sequences at time zero, 60 minutes, and 120 minutes into the test. The duration of these startups lasted well under one minute including the spin down process. At the conclusion of this test, the motor appeared to exhibit the symptoms of an internal winding short and was unable to be restarted after minor repairs were attempted.

Motor 22, the second of these motors, was run continuously throughout the test with no intermittent startups unlike the first motor. Similar to the first motor, smoke began to be emitted at just under 40 minutes into the test. This test concluded at just over two hours with a single phase shorting within the motor resulting in an inability to produce the demanded load. These two initial tests indicate that this method of aging would be much more suited to short term data collection of electrical failures for mid scale motor systems. Additional follow-on work can use this simplified test plan to rapidly generate viable experimental data to develop and validate processing codes with.

8.2 Large Neoprene Impeller Degradation Experiment

One of the degradation data collection experiments in the University of Tennessee's laboratories that were utilized in this project involves the accelerated degradation of multiple neoprene impellers in small-scale horizontal pumps. The pump contains a six bladed neoprene impeller, which has been subjected to high levels of heat stress. The aged impeller is then placed in the pump in order to determine the time to failure. The parameters that are monitored during testing

are vibration, differential pressure, and current of the pump. To determine the thermal degradation point of the neoprene impeller, a literature review was conducted to examine past work in the thermal aging of neoprene. The thermal testing methods for the impellers, the test bed facility, and data collection methods, as well as the initial findings of the data analysis are presented in this report.

8.2.1 Literature Review

To develop a testing procedure for the thermal aging of the neoprene impellers, a literature review into past studies of thermal aging of polychloroprene are investigated to determine the thermal degradation point. Also, since the point of the experiment is to degrade the impeller, it was also informative to review past literature on the stress-strain relationships of neoprene as it is thermally aged.

8.2.1.1 Thermal Aging of Polychloroprene

In this section, we investigate some of the accelerated thermal aging tests performed on pure neoprene, neoprene compounds, and vulcanized neoprene. Pure neoprene rubber consists of hydrocarbons, CH and CH2 linked with chlorine. Sulfur and other metallic compounds can be added to the neoprene to give vulcanized rubber or a neoprene compound [Gardner 1971]. For pure neoprene rubber, the thermal degradation point is given as 50 C [Ramuhalli 2012]. For neoprene compounds or vulcanized neoprene, the thermal degradation point can range from 100-615 C [Dadvand 2000, Gardner 1971, Gillen 2005, Budrugeac 1990]. This wide range of temperatures for the degradation point depends on the fillers and additives added to the neoprene. This type of rubber does not have a true melting point as other polymers but rather goes through a degradation period that involves reduction of the weight of the sample by release of hydrogen chloride, this occurs since chlorine is a major constituent of neoprene. As the sample is heated to higher temperatures, the material decomposes and loses structure. The danger involved when heating this material to these high temperatures is that the release of volatiles such as chlorine and hydrogen chloride gas can be quite toxic. Figure 8.2-1 shows the percent of weight loss of two neoprene samples as the temperature is increased.

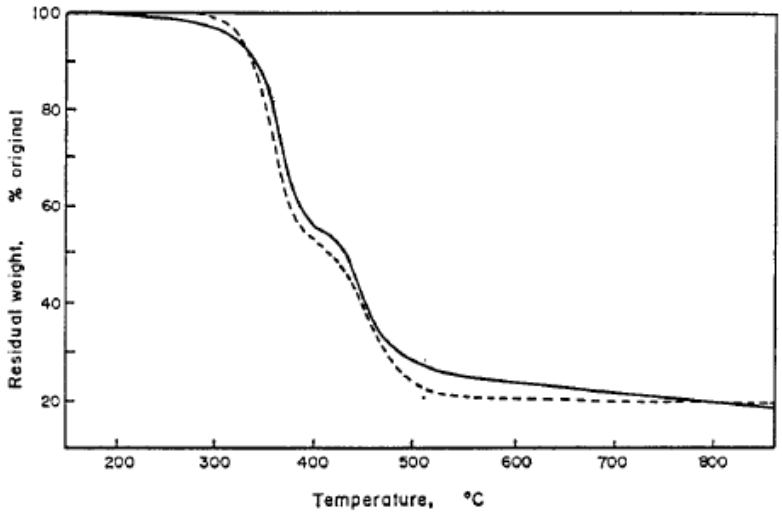


Figure 8.2-1 Percent Weight Loss vs. Temperature for Two Neoprene Compounds [Gardner 1971]

Based on these findings, it appears that to achieve substantial degradation, the neoprene should be exposed to temperatures in excess of 350 C. Much of the remaining available literature focused on thermal aging the neoprene rubber at temperatures of 80-140 C for extended periods of time and then looking at any chemical or structural changes. In one such test, a neoprene compound was aged at 80 C for three months and chemical analysis was performed on the aged and pure sample in order to determine if there were any significant changes in the chemical makeup of the rubber [Dadvand 2000]. Figure 8.2-2 shows the pyrolysis results at 387 C for each of the samples.

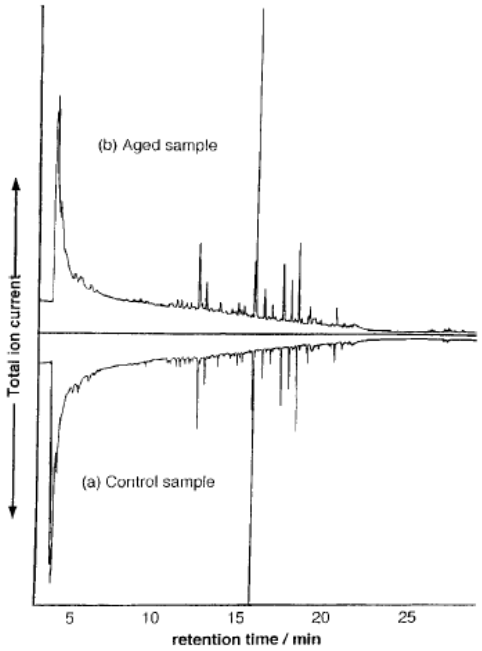


Figure 8.2-2 Pyrolsis Results for Control & Sample Aged 3 Months at 80 C [Dadvand 2000]

Looking at Figure 8.2-2 it is seen that the chemical peaks of each sample were virtually unchanged for each of the samples. This suggests that thermally aging at 80 C for extended periods of time would not be beneficial for this research. Finally, other studies examined the non-Arrhenius behavior of the neoprene during thermal aging tests [Gardner 1971]. The Arrhenius relationship is best described as a linear relationship between temperature and remaining life of a component. The relationship states that for every 10 C rise in temperature, the remaining life of a component decreases by a factor of 2. In most cases, this relationship takes on a linear form, but in the case of the thermal aging of neoprene, there is a noticeable curvature in the resulting plots. The form of the Arrhenius relationship used in thermal degradation studies takes on the form [Budrugeac 1990]:

$$k(T) = A\exp(-\frac{E_a}{RT})$$

Equation 8-1

Here, k(T) is a function for the constant rate of degradation, A is a constant, E is the activation energy, both based on the material properties, R the universal gas constant and T the absolute temperature. Figure 8.2-3 shows the Arrhenius plot of a neoprene sample plotted with the activation energy.

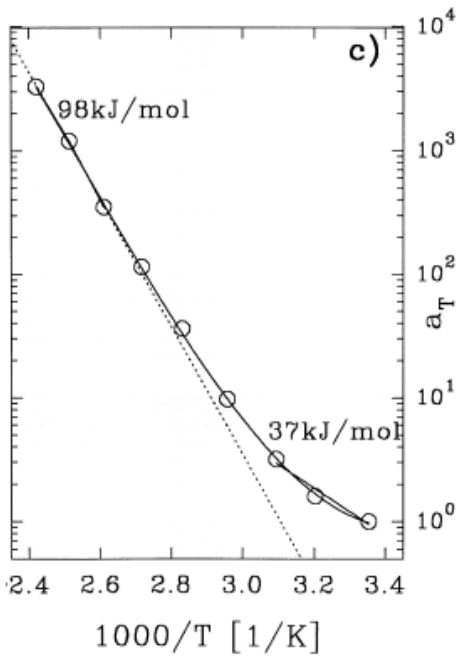


Figure 8.2-3 Arrhenius Plot of Neoprene Sample with Activation Energy [Celina 2000]

The reason for the curvature in the Arrhenius plots that as the temperature and time of aging is increased, the activation energy of the rubber is of a lower value at higher temperatures than at lower temperatures. This means that it takes more energy at lower temperatures to degrade the neoprene, since the activation energy is higher, which occurs from oxidation of the samples when heated in air. Next, we will look at the mechanical changes of neoprene after thermal aging.

8.2.1.2 Mechanical Changes after Thermal Testing

In this section we examine some research into the mechanical changes in neoprene after thermal aging. When the rubber is exposed to temperatures above 100 C, cross linking of the chlorine and other compounds in the rubber stiffens the structure. This makes the rubber less flexible and can tend to break when exposed to stress or pressure, such as during operation of the pump. In one study, neoprene glove swatches were first soaked in acetone and then thermal aged at 100 C for 16 hours. After aging, the tensile strength of the gloves was tested in order to determine if there was any change in the property of the rubber [Gillen 2005]. Figure 8.2-4 shows the results of the tensile tests after 10 of these cycles.

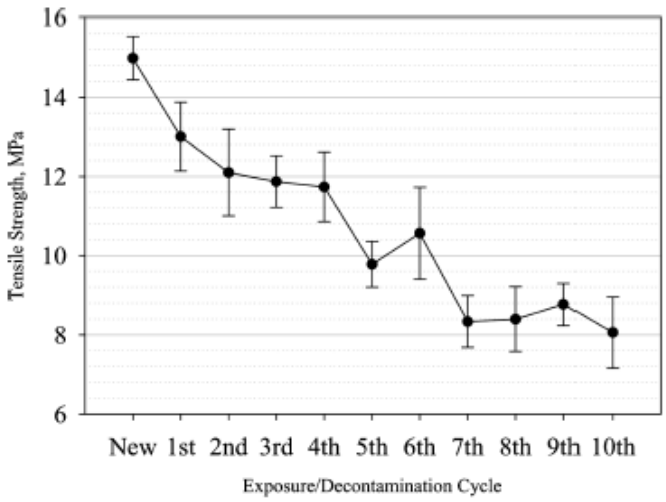


Figure 8.2-4 Change in Tensile Strength of Neoprene after Acetone Exposure and Thermal Aging [Gao 2007]

Looking at Figure 8.2-4, it is seen that the tensile strength of the gloves reduces from 15 MPa to 12 after four cycles, and after ten cycles reduces to 8 MPa. It is noted that the thickness of the neoprene swatches was only .75 mm, while the neoprene impellers used in the study are approximately 3.2 mm thick. This study does show that the neoprene material will stiffen after repeated thermal aging and this stiffening would help degrade the impellers during experimentation. Another study looked at thermally aging neoprene sheets 1.6 mm thick at 120 C for 24-168 hours and calculating a stress-strain curve relationship by testing the elongation and tensile strength of the samples [Budrugeac 1990]. Figure 8.2-5 shows the results of this study.

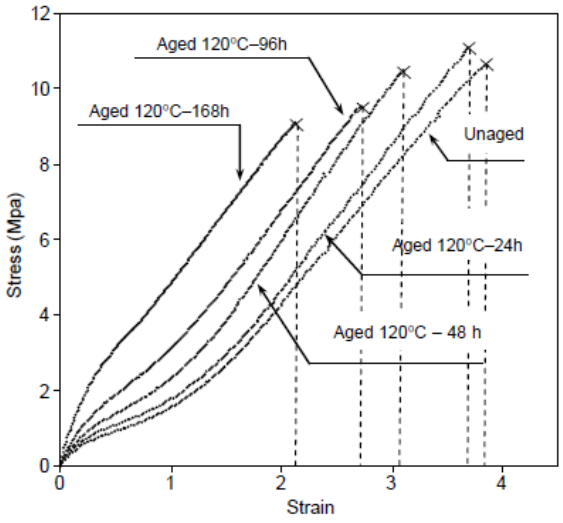


Figure 8.2-5 Stress-Strain Curves of Neoprene after Thermal Aging [Celina 2000]

Looking at Figure 8.2-5, it is seen that there is virtually no change in the stress-strain relationship of samples aged for 24 hours when compared with the results for an unaged sample. At 48 hours of aging, the strain values of the samples start to show a decrease but the stress values are virtually the same as an unaged sample. After 96 and 168 hours of aging, the strain values for these samples are almost half of that for the unaged and the stress value decreases from 11 MPa to around 9 MPa. The results of the figure can be interpreted as an increase in the formation of cross-linking in the samples during aging. For a low cross-link density, the rubber remains soft and flexible and as the density of the cross linking increases, the structure of the rubber becomes more rigid or inflexible. This can cause the structure of the rubber to snap or break apart when pressure is applied to the sample [Celina 2000, Dadvand 2000, and Budrugeac 1990]. Based on these findings and that in the previous section, it was determined that testing of the impellers to find the thermal degradation point would be carried at temperatures of 160-280 C and then determine which temperature offered the correct amount of degradation that the impeller could still be used in experimentation but not immediately fail.

8.2.2 Physical Setup

This section describes the small-scale proof of concept experiment. This experiment focuses on thermally aging the neoprene impellers used in small-scale pumps and then running the pumps to failure. Here, failure constitutes the inability of the pumps to draw any water from a common sump. The test bed for the accelerated life testing of the neoprene impellers consists of four small-scale 115 Volt transfer pumps mounted six feet above the common sump, which contains only water. Figure 8.2-6 shows the test bed facility used in the experiment.

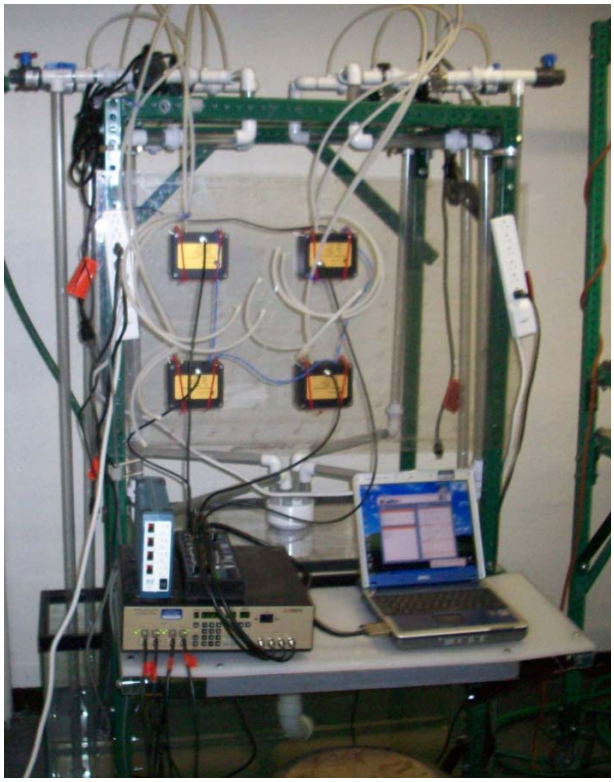


Figure 8.2-6 Experimental Test Bed Setup

The signals measured from the pumps include current drawn by the pump, which is measured by a standard current clamp attached to the power cord of the pump. A differential pressure transducer measures the differential pressure across the inlet and outlet of the pump. Finally, the vibration is measured by an accelerometer that is mounted to the pump housing. The sampling frequency of the signals is 1024 Hz and data is recorded for two seconds every ten minutes. A listing of the components used is shown in Table 8-4.

Table 8-4 Components Used in Testing

Use	Manufacturer	Name	Model
Degraded Unit	Jabsco	Self-Priming Transfer Pump	115 Volt AC PC2
Current	Fluke	Current Clamp	I200s
Accelerometer	PCB Piezotronics	ICP Accelerometer	YJM352C18
Pressure Transmitter	GE	Pressure Transducer	M4020ED01R

8.2.3 Thermal Aging Procedures:

It was found during preliminary aging tests that many of the temperatures listed for the degradation of neoprene did not apply for this particular impeller material. Impellers were first

heated at 45, 80, 100 and 140 C in an air circulated oven for upwards of 6 hours and then placed into the pumps. It was found that using these temperature ranges that there was no appreciably degradation of the impellers when used in the pumps. Since the impeller material is vulcanized, the impellers were next heated at temperatures of 240, 260 and 290 C for upwards of 6 hours. It was found that using these high temperatures severely stiffened the material and the blades of the impellers snapped when slight pressure was applied. Next, temperatures ranging from 150 C to 190 C were used to determine which would cause some degradation but have the impeller remain flexible enough for operation in the pump. The optimum temperature that was settled on was 165 C and an aging time of 1.5 hours. This temperature and time caused some degradation but not so much that the impeller would fail prematurely. Typically, the impellers lasted from 1-5 days but what was also found was that some impellers lasted 3-7 weeks. Future tests with this type of material would need to have the aging temperature or time increased so that the slower degradation mode would not appear and the range of failure would be more consistent

8.3 Heat Exchanger Fouling Experiment

This test bed experiment was designed in an effort to validate the PEM and PEP toolboxes effectiveness in developing prognostic models. When designing the heat exchanger, the goal was to generate data that effectively represents time-series data with a reliable degradation mechanism as well as accurately represent data from a commercial heat exchanger. The accelerated test bed was designed with the goal of producing degradation data sets in short periods of time in comparison to true reactor heat exchanger maintenance schedules, which can vary from 6 to 18 months on average. The purpose of the test bed data was to validate functions that select prominent features, develop models trained on unfaulted data that produce trending residuals, optimally combine residuals into a prognostic parameter, calculate RUL and corresponding uncertainty, and make maintenance schedule decisions based on these calculations.

8.3.1 Experimental Setup

The design of the accelerated heat exchanger test bed was developed to produce data similar to that from a commercial heat exchanger. Early accelerated test bed designs were chosen to produce data over a 14-day cycle compared to maintenance schedule of roughly a year for commercial (reactor) heat exchangers. The heat exchanger chosen for accelerated degradation application is a 64-tube counter-flow shell-and-tube BASCO heat exchanger. To facilitate the failure mechanism, kaolin clay is added to the heat exchanger hot leg, which expedites particulate fouling on the heat exchanger tube walls. At the beginning of each cycle, 105g of clay are added to initialize the clay/water mixture, and then each subsequent 48 hours an additional 75g of clay

are added. This keeps a consistent concentration throughout the operating cycle. To capture the data, several transducers are placed at vital points along the heat exchanger setup. The failure mechanism of the heat exchanger that this project is trying to detect is particle fouling. When clay particles accumulate on the walls of the heat exchanger, the heat transfer between the hot and cold legs is impaired. In order to determine the heat transfer for the heat exchanger, the following must be measured: hot leg inlet temperature, hot leg outlet temperature, cold leg inlet temperature, cold leg outlet temperature, hot leg flow rate, cold leg flow rate. The inlet and outlet pressure of the hot leg is also measured in order to manipulate the flow rate to achieve consistent cycles. There are also four temperature sensors placed throughout the heat exchanger tubes, which measure the temperature gradient across the heat exchanger. A plot of the 12 sensors for the heat exchanger setup is shown in Figure 8.3-1.

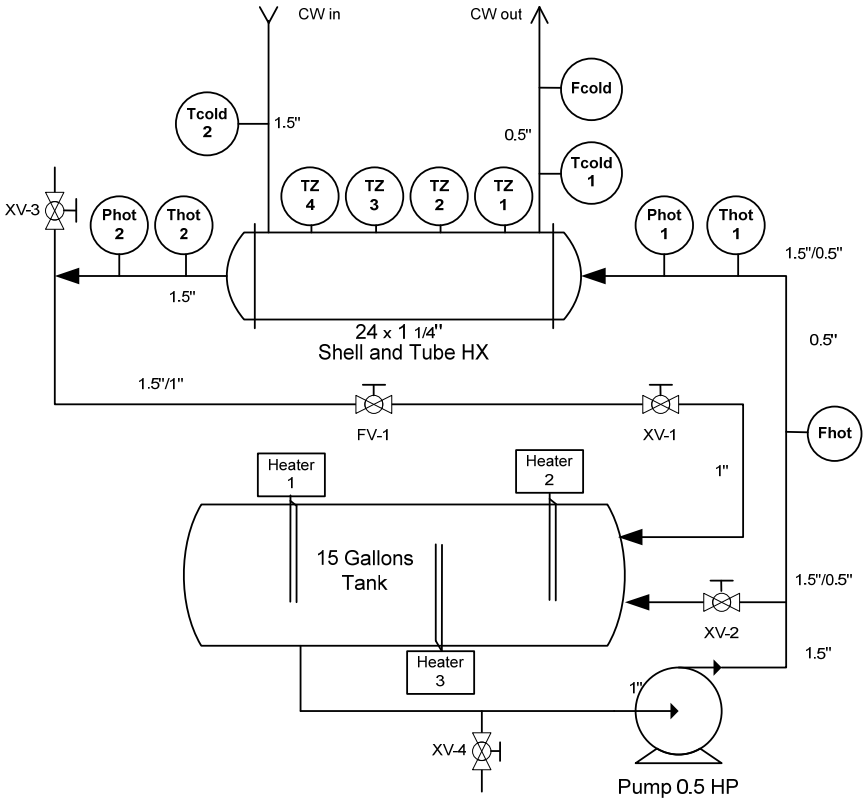


Figure 8.3-1 Schematic of heat exchanger physical setup

To heat the water in the hot leg, there are three 250W heaters placed within the test bed's 15-gallon tank as seen above. To force flow through the heat exchanger, a 0.5HP pump is placed at the bottom of the system. The flow rate is controlled by two valves; a ball valve (XV-2) for rough flow control, and a needle valve (FV-1) for fine flow control. The clay is added every 48 hours through the port at valve XV-3.

Three National Instruments modules are used to read the transducer values from the test bed. The voltage signals from the data acquisition board are processed using a LabVIEW program shown in Figure 8.3-2.

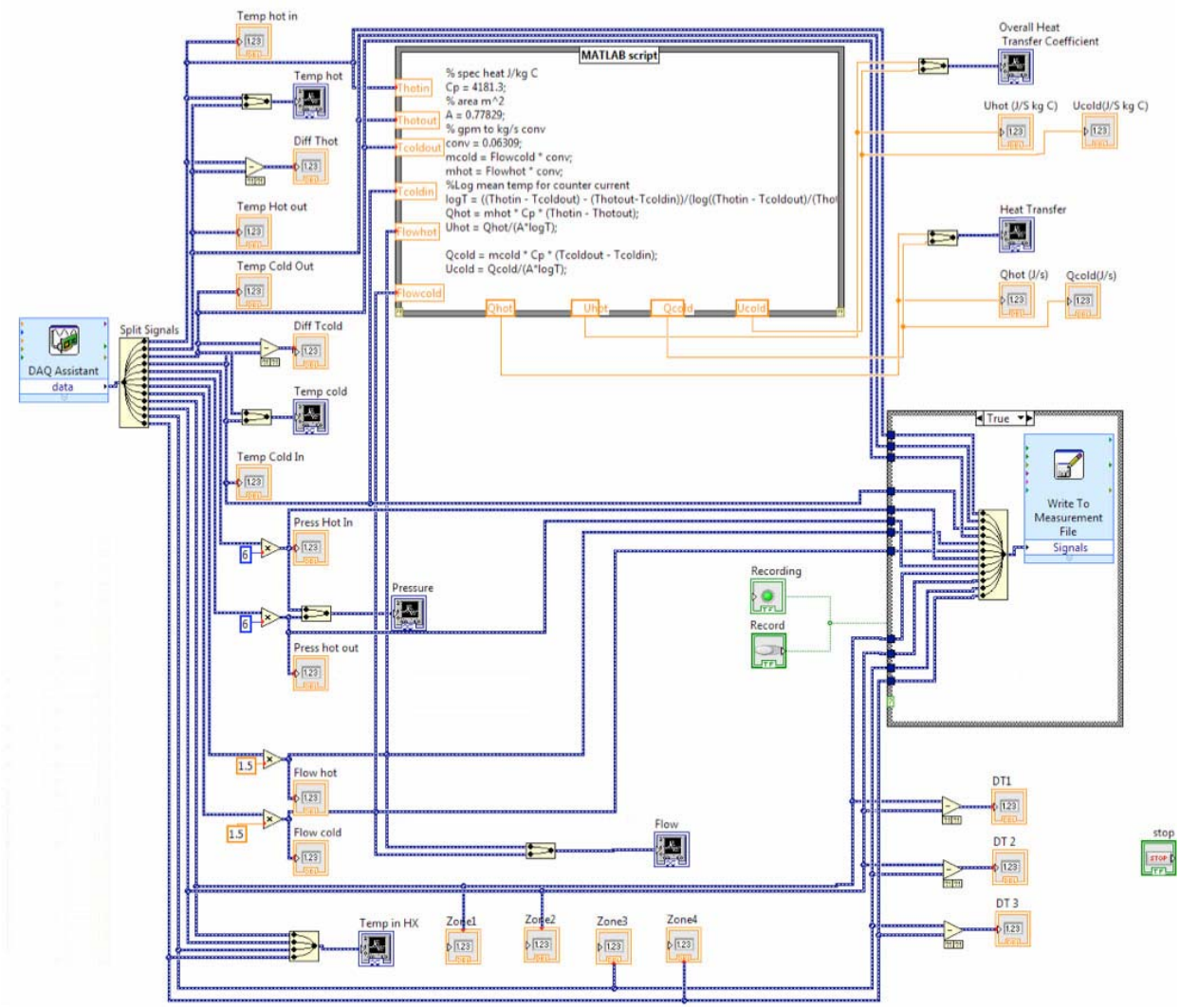


Figure 8.3-2 View of program structure for data acquisition using LabVIEW software

This program outputs the 12 signal values to an excel file, which is then extracted into a MATLAB file using the function in Appendix 14.3. This data can then be processed for prognostic model development.

8.3.2 Experiment Modifications for Improved Degradation Cycles

After several cycles were run, a few modifications were considered in order to improve the consistency of each cycle, as well as remove artifacts within the data that could potentially harm future modeling efforts. Early designs had the heat exchanger running in same-flow direction,

meaning that both the hot and cold legs ran from the right side of the heat exchanger to the left. This was quickly changed to counter-flow operation, which had a more desirable physical model, as well as produced better observable degradation in the data. Another change that was made to the system was the addition of 75g of clay every 48 hours after starting the heat exchanger. Originally, the 105g at the beginning of each cycle was the only clay added. It was found that over the 14-day cycle, some of the clay would adhere to the walls of the heating tank. In order to maintain a relatively constant clay/water concentration, the additional clay amount was chosen. Similarly, another modification to the system was the addition of a ball valve on the return hot leg, which could be opened to exhaust a small amount of clay/water mixture in order to visually inspect it for concentration. This modification is shown in Figure 8.3-3.

Figure 8.3-3 Drain modification to heat exchanger for clay concentration sampling

Currently, this addition allows for visual inspection to determine whether or not the clay concentration remains constant throughout the cycle, and is achieved by comparing the test sample to known concentration cases. For the time being, visual inspection has proven to be adequate in the absence of more costly and time-consuming quantitative tests. In Figure 8.3-4 is an example of visual inspection.

Figure 8.3-4 Visual check of clay/water concentration through a cycle

The goal of the visual inspection is to make sure there is no noticeable change in concentration over a cycle. The last modification to the heat exchanger experiment was made to the procedure when adding clay. Originally, every 48 hours when the clay was to be added to the system, the procedure was as follows: turn off pump, open feed-valve, add clay/water mixture, close feed-valve, turn on pump, slowly open valve to bleed air out of the system. The results of this process over a 14-day cycle are shown in Figure 8.3-5.

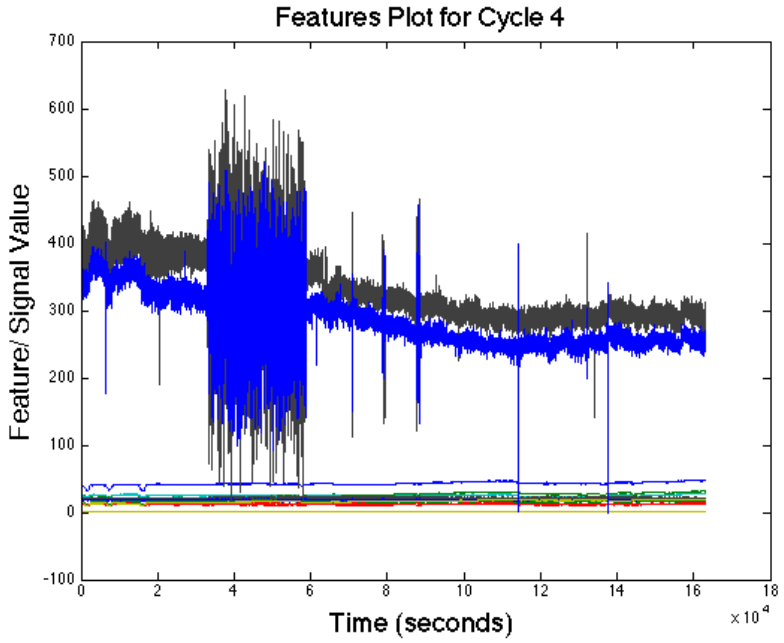


Figure 8.3-5 Plot of features and signals for a single cycle (cycle 4) without filtering or data acquisition modification

In the figure above, it can be seen that when the system is opened to add clay, there are large spikes in the data. For some cycles, the system was thrown into a transient for several thousand observations (in cycle 4 above this corresponds to observations ~35000 to ~60000). These transient locations, as well as each data spike, significantly impaired modeling efforts. To reduce these artifacts in the data, the procedure for adding clay was slightly modified. Rather than leaving the data acquisition software on (recording) while the clay was added, the procedure was changed and the data acquisition was paused while the valve was open and the pump was off. After the clay is added, the valve closed, the pump turned back on, and the system air bled off, and then the data acquisition software is turned back on effectively removing each data spike. The results of a cycle with this new procedure are shown in Figure 8.3-6.

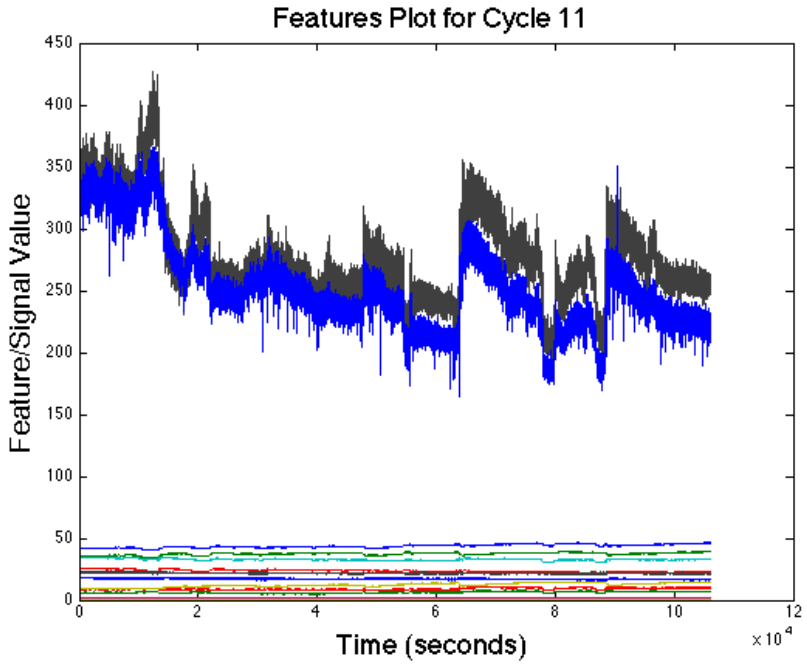


Figure 8.3-6 Plot of features and signals for a single cycle (cycle 11) when data acquisition modification is implemented

As the figure shows, there are far fewer large spikes in the data. It can still be seen where the system is opened, but the models that result from the cycles without the spikes are improved. A further procedural modification to note is the addition of two operational days at the beginning of each cycle to gather unfaulted data, when no clay (including the initial 105g) has been added to the system. This creates training data with no clay fouling to influence the data. This modification has only been recently implemented, and is not present in the models discussed in future sections of this report.

8.3.3 Status of Experiment/Operating Conditions

There are 16 completed heat exchanger cycles. 8 cycles were completed at 1 gallon-per-minute operating in the cross-flow direction. One cycle was completed at 1 gallon-per-minute operating in the same-flow direction (for comparison). 7 cycles have been completed at 1.5 gallons-per-minute operating in the cross-flow direction. The different operating conditions allow for Type II prognostic models to be developed, as well as increased Type III model robustness, which will be discussed in depth in future sections. There is an additional 1.5 gallons-per-minute cycle to be run starting next week, and is predicted to be complete before the end of this project resulting in a total of 8 cycles at 1 gallon-per-minute, and 8 cycles at 1.5 gallons-per-minute. After making several repairs to the heat exchanger through the course of this project, the heat exchanger is still in working condition and ready for additional future cycles if necessary.

8.3.4 Data Pre-processing

In current modeling efforts, there is not a dedicated unfaulted data set. In order to train a model, the first 10 hours of operating data is assumed to be unfaulted, and is cut from the faulted data for each cycle. Both the unfaulted and faulted data sets are then passed through a series of filters and manual cleaning steps in order to remove irrelevant artifacts in the data, as well as prepare the data for down sampling. By using median filters, information contained in observations that are removed during down sampling can be merged into the remaining data. An example of the filtering method applied to the faulted data is the hot leg flow rate signal shown in Figure 8.3-7 before and Figure 8.3-8 after pre-processing.

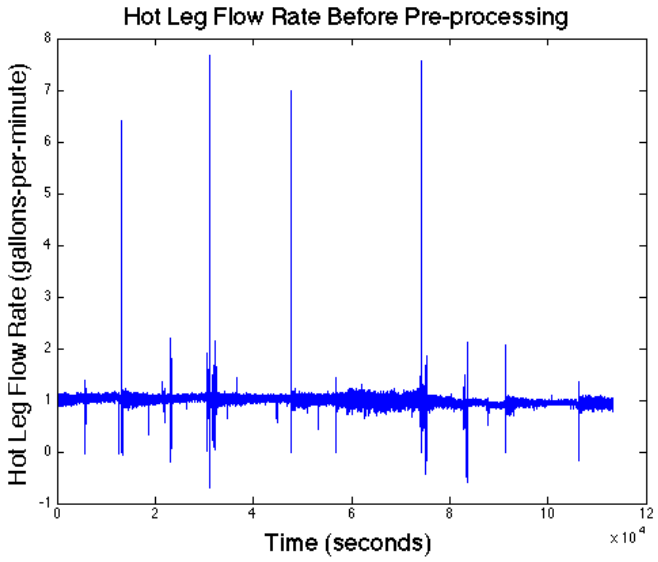


Figure 8.3-7 Example of the hot leg flow rate before pre-processing

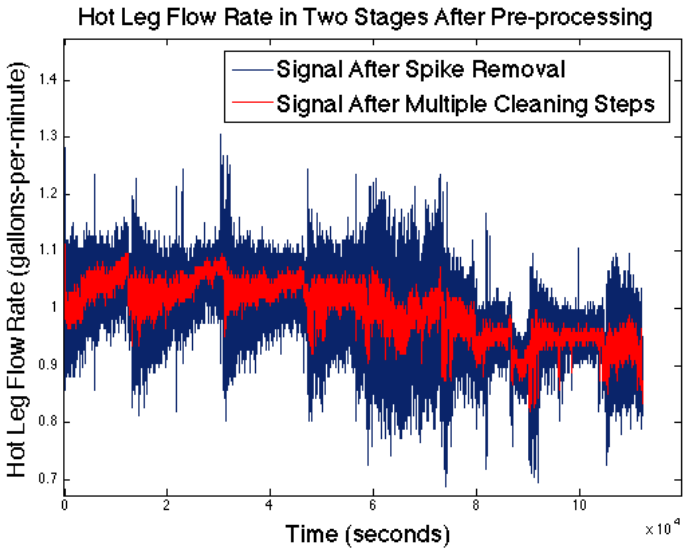


Figure 8.3-8 Example of the hot leg flow rate after stage one of pre-processing (spike removal - blue) and stage two of pre-processing (filtering and cleaning - red)

The faulted cycles are then cut to remove excess observations; the training, testing and validation data is generated using vector selection function from the PEM toolbox. At this point, the data is ready for modeling.

8.3.5 Experimental Results

Each completed cycle includes data for the following sensors: temperature of the hot log in, temperature of the hot log out, temperature of the cold log in, temperature of the cold log out, mass flow rate out of the hot side, mass flow rate out of the cold side, pressure of hot leg in, pressure of cold leg out. The data from cycle 6 is plotted in Figure 8.3-9.

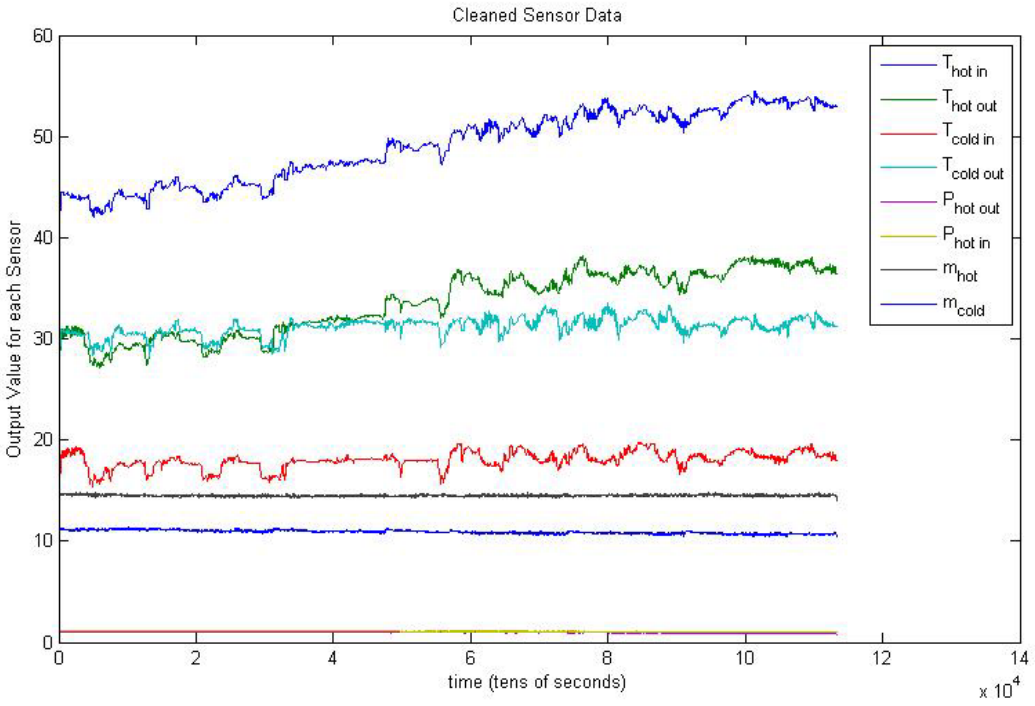


Figure 8.3-9 Plot of signals after cleaning for a single cycle (cycle 6)

A plot of example features is included in Figure 14.1-1 following this section of the report. The degradation due to fouling can be observed in the increased temperatures across several signals over time.

8.4 Passive Material Test Bed

PNNL has recently developed a set of passive component (materials) degradation assessment test-beds. The objective of these test beds is to enable in-situ measurement of materials degradation using multiple nondestructive evaluation (NDE) methods. The measurement data from these test-beds is intended to assess two key questions: (1) can advanced NDE methods be used to identify degradation precursors (i.e., signatures that are sensitive to materials micro structural changes that are indicative of damage accumulation prior to the onset of visible cracks) and (2) can advanced NDE measurements of precursor states be used to predict remaining useful life of the material or component, under typical stress conditions?

This section briefly discusses the different NDE methods that are being utilized, as well as the test beds themselves. NDE measurements being considered for this project include ultra sonic transmissions, magnetic Barkhausen noise, and acoustic emission.

8.4.1 Nondestructive Evaluation Measurements

The challenges associated with characterization of aging in NPP materials are significant (Doctor 1988; Dobmann 2006). Current NDE techniques used for NPP in-service inspection (ISI) to detect materials degradation are typically applied to detection of large flaws that occur near the end of component life. However, recent years have seen a move towards NDE for early damage detection in NPP materials (Bond et al. 2009, 2011). Monitoring for early detection of materials degradation requires novel sensors and enhanced data integration techniques. A range of acoustic and electromagnetic measurement methods may be suitable, including non-linear acoustics (Cantrell and Yost 2001; Ogi et al. 2001), eddy current (Lois and Ruch 2006; Ramuhalli et al. 2010) and magnetic Barkhausen emission (Raj et al. 2003; Dobmann 2006). However, the sensitivity of these techniques to precursors from damage mechanisms of interest to LWR long-term operations is not clear, and needs to be quantified. Further, there are still no accepted measurement technologies for the detection and assessment of some degradation mechanisms unique to NPPs, such as void swelling.

The magnetic Barkhausen effect is a result of the magnetic hysteresis of ferromagnetic materials (Jiles 2000). The magnetic flux density in ferromagnetic materials placed in an external applied magnetic field is a function of the applied magnetic field, with larger numbers of magnetic domains within the material aligning with the applied field direction with increasing applied field strength. This realignment is, however, not a continuous process, since the presence of dislocations or other damage pre-cursors results in domain wall pinning. Increasing the applied field strength results in abrupt realignment of some domains, and is accompanied by a release of energy that may be detected using a sensing coil.

Like all electromagnetic methods, the magnetic Barkhausen method is predominantly a near-surface measurement, with the standard depth of penetration (the distance into the material where the induced current density decreases to 37% of its value at the surface) decreasing with increasing frequency (ASNT 2004) defined as:

$$\delta = \sqrt{\frac{1}{\pi f \mu \sigma}} \quad \delta = \sqrt{\frac{1}{\pi f \mu \sigma}}$$

Equation 8-2

where f is the excitation frequency, μ is the magnetic permeability of the material, and σ is the electrical conductivity. For non-ferritic steel (such as 304 or 316L), the skin depth at 1 kHz is about 13.1 mm. In many stainless steels, the effect of increasing damage (through mechanisms

such as fatigue loading) is an increase in dislocation density. At the same time, in certain steels, damage can result in conversion of austenite to a ferritic phase. The impact of these changes is two-fold, resulting in local changes in conductivity and permeability. These phenomena combine to impact the Barkhausen noise measurement from steels subjected to aging and degradation. However, the correlation between the measured parameters and the amount of damage is not linear, and is a function of several other variables (such as hardness). The Barkhausen noise measurement method has been applied to determine residual stresses in ferritic steels, and to determine the amount of hardening or cold work. Studies have also shown that this technique is sensitive to damage precursors in ferromagnetic materials, and quantities such as the energy and peak value in the Barkhausen signal have been shown to correlate well with level of damage in materials (Parakka et al. 1997; Gorkunov et al. 2000; Sullivan et al. 2004; Sagar et al. 2005; Hakan Gur and Cam 2007).

A detailed history and introduction to acoustic emission testing is provided in the American Society of Nondestructive Testing (ASNT) Handbook (ASNT 2005). Fundamentally, acoustic emission (AE) is the elastic energy released during deformation of materials (ASNT 2005). The released energy travels as a transient elastic wave in the material and is typically recorded using a transducer that is located at some distance from the AE source. In metals, several phenomena give rise to AE, including crack initiation and growth, phase transformations, twinning, deformation, etc. Factors such as leaks also give rise to changes in the local stress gradients, resulting in a transient elastic wave.

8.4.2 Bench-Scale Test Beds

There are several bench-scale test-beds under development at PNNL that are related to the onset of fatigue cracking. The first of the test setups is to study thermal fatigue while the second is to study mechanical fatigue. In both cases, stainless steels (304 grade and 410 grade) are used initially.

The thermal fatigue bench-scale setup uses tubular specimens, Figure 8.4-1, to enable relatively easy online nondestructive monitoring. The tubular specimens are heated from the inside with periodic cooling on the outside. Heating is accomplished by means of a cartridge heater on the inside of the hollow specimen, Figure 8.4-1, while cooling using the water sprays was from the outside. The resulting thermal stresses initiate a thermal fatigue crack on the outside surface of the hollow specimen, providing easy access for both online (either continuously, or in interrupted test mode) and offline (i.e., by removing the specimen from the test setup) NDE measurements as well as planned destructive analysis of the specimens.

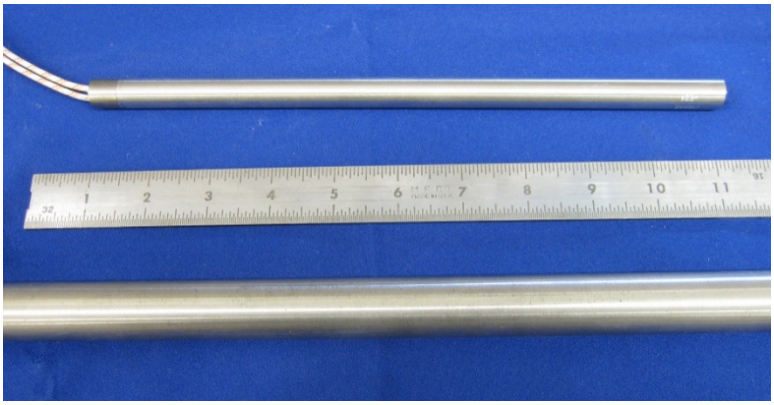


Figure 8.4-1 Example of Tubular 304SS Specimen and Cartridge Heater

Two test stations with hollow 304 stainless steel (SS) rods are available (Figure 8.4-1). At each station, one specimen is heated to temperatures up to 600°C and cooled to temperatures as low as 30°C by a periodic water spray controlled by a timed solenoid valve arrangement. Figure 8.4-2 presents a snapshot of the fatigue process in operation at both stations, with one specimen (upper) just starting the heating cycle while the other (Figure 8.4-3) just starting the cooling cycle. Thermocouples are used to monitor and control the heating and cooling cycles. The thermal cycles for these runs are set to 4–5 seconds of water cooling and a total cycle length of 50 seconds, or 72 cycles per hour. This amount of cooling allows the sample temperature to drop from about 600°C to 30°C in the center portion of the heated rod for a maximum thermal fatigue stress.

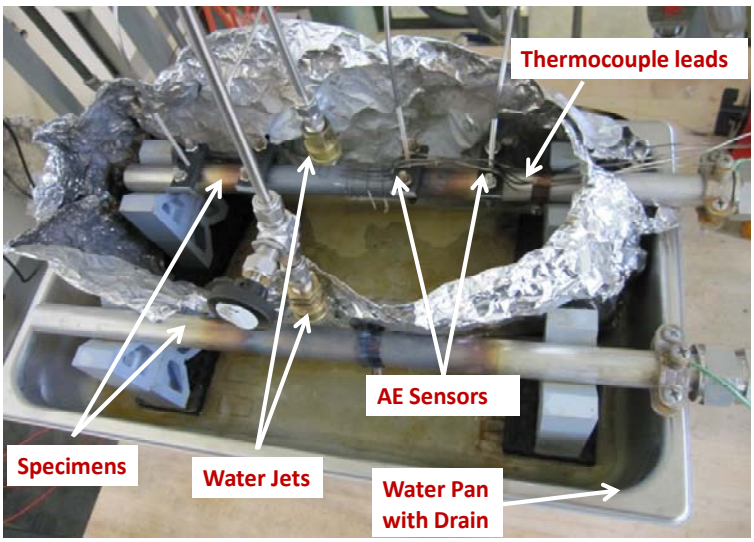


Figure 8.4-2 Thermal Fatigue Setup with Two Stations. Tubes are heated and cooled independently at each station. The setup shown has two water jets that impinge on the top of the heated specimens, assorted control and recording thermocouples, and acoustic emission sensors attached with clamps to the upper sample.

Figure 8.4-3 Thermal Fatigue Setup Operation. The upper sample has cooled below the visible thermal temperature while the lower sample has just started the cooling cycle and is still red hot. The shape of the water jets can be seen in this photo, most clearly for the upper sample.

The mechanical damage setup uses specimens of 410-grade stainless steel in an interrupted tensile experiment. The specimens are ASTM-standard tensile test specimens, with a gauge length of 6 inches (152.4 mm) and specimen thickness of 0.375 inches (9.5 mm). The specimens are annealed prior to the tensile test to ensure that all samples have the same initial stress state. The specimen is held in an Instron MTS machine (Figure 8.4-4) and strained in increments of approximately 2%. After each test, the stress is released and measurements made at multiple (typically 2-3) locations in the gauge region on the specimen. The locations are generally selected to be symmetric about the center of the specimen. The specimen design and applied loading is such that the strain in the specimen is expected to be uniform along its gauge length (at least until damage localization and necking of the specimen occurs). As a result, the expectation is that the measured data at the selected locations would be similar (within measurement noise) until damage localization occurs, at which point the measurements at the different locations should start to deviate. At each location, multiple measurements are typically made to assess repeatability and quantify measurement noise levels. Probe placement is done manually, with the probe lifted away from the surface between successive measurements. The specimen is then placed in tension, strained by an additional 2%, and released. Measurements are again made at the same locations. This process was repeated until the specimen failed.

TOP

BOTTOM

(a) (b)

Figure 8.4-4 (a) Experimental setup for tensile test. (b) Typical measurement locations on tensile specimen.

8.5 Offsite Bearing Testing Facilities

The University of Tennessee also collaberated with the AMS Corporation to collect and share bearing degradation data that was used for prognostic modeling. This data will be used to verify and validate the technologies and methodologies developed during this project. The AMS test bed, shown below in Figure 8.5-1, consists of a steel frame with two welded bearing mounts to support the test bearings.

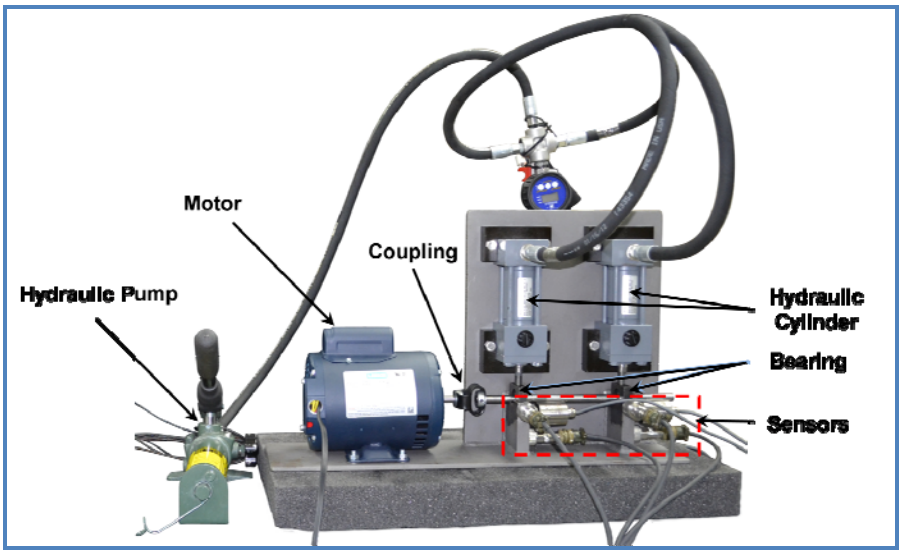


Figure 8.5-1 AMS Bearing Test Bed

A hydraulic cylinder is mounted directly above each bearing and pressurized by a hydraulic hand pump to apply a large radial force and accelerate the degradation rate. A base-mounted AC motor rotates the steel shaft inside each bearing at 3450 revolutions per minute. Accelerometers (IMI 603C01) and ultrasonic sensors (Ultra-Trak 750) are magnetically attached to each mount to monitor sonic and ultrasonic vibration of each bearing. Ultrasonic sensors are used in this test bed to evaluate if they can provide an earlier indication of bearing degradation than the accelerometers.

8.5.1 Experiment Setup

The experiment at AMS focused on daily application of a predetermined load to a bearing, and then running this bearing to failure, with the goal of trying to determine how much life a typical bearing has remaining after a load is applied. The only measurements taken during the experiment were vibration and audio signals.

Data was collected using a sampling frequency of 12500 Hz. During the course of the experiment 30 bearings were tested to failure, with failure times ranging from 5-500 tests. Investigation of the supplied data sets showed that some bearings were tested and had measurements from eight signals while other bearings only had measurements from four signals. Table 8-5 lists the signals recorded for the each of the previously mentioned cases.

Table 8-5 Bearing Experiment Measured Signals

Case/Signal Number	8 Signal Cases	4 Signal Cases
1	Horizontal Inner Vibration	Horizontal Inner Vibration
2	Axial Inner Vibration	Axial Inner Vibration
3	Horizontal Outer Vibration	Horizontal Outer Vibration
4	Axial Outer Vibration	Axial Outer Vibration
5	USLVL Inner Vibration	
6	USLVL Outer Vibration	
7	US Audio Inner	
8	US Audio Outer	

It is unclear why some bearings were measured with fewer signals, so each of the cases will have to be analyzed separately.

8.5.2 Data Analysis

In this section, some of the preliminary results are shown involving the extraction of useful features from the supplied data in both the time and frequency domains. As mentioned before, 30 bearings were run to failure during the course of the experiment, with a range of failure times occurring between 5-500 tests. The data was sampled at 12500 Hz and each of the separate tests had 20972 observations. Emphasis was placed on bearings that had a large number of tests before failure, since bearings that have failed quickly will not be very useful for modeling as trends indicative of impending failure will not typically manifest in the time or frequency domains. Therefore these bearings will not be considered during analysis. The goal of the data analysis is to extract a set of useful features that show trends that tend to failure. Ideally, all of the bearings considered should have the same functional trend so that useful prognostic parameters and models can be developed. First, some results in feature extraction in the time domain are shown, and next the analysis of the frequency domain will be provided.

Time Domain Preliminary Results

A useful method for extracting feature information in the time domain from periodic signals, such as those found in rotating machinery, involves calculating the statistical moments in a windowed fashion through the course of testing. This procedure is the same one that was used in the motor degradation data analysis. For this current project, 2 data points were extracted from each of the tests for each of the bearings considered. As an example, the first bearing file had 87 total files, and so for each statistic there are a total of 174 data points. Many of the data files contained a sensor bias of 10 Hz which was removed during processing in order to mean-center all signals so that the bias would not carry over into further analysis.

An example of an unusable extracted feature is shown in Figure 8.5-2. The feature shown is the mean of the signals in the first bearing file.

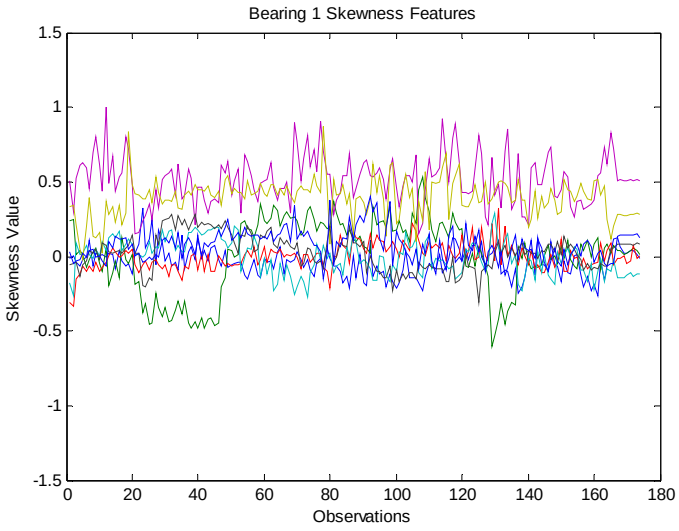


Figure 8.5-2: Bearing 1 Feature Plot: Skewness

Looking at the figure, it can be seen that the skewness values for bearing 1 do not show any type of trend as testing progressed. Rather, all of the signals show a fairly constant value. One thing to consider is to track signals that show a positive or negative skewness value as testing progresses. During testing, Signals 5 and 6, which are the vibration measurements, show positive skewness while other signals do not.. Next, the kurtosis values for bearing 1 are shown in Figure 8.5-3.

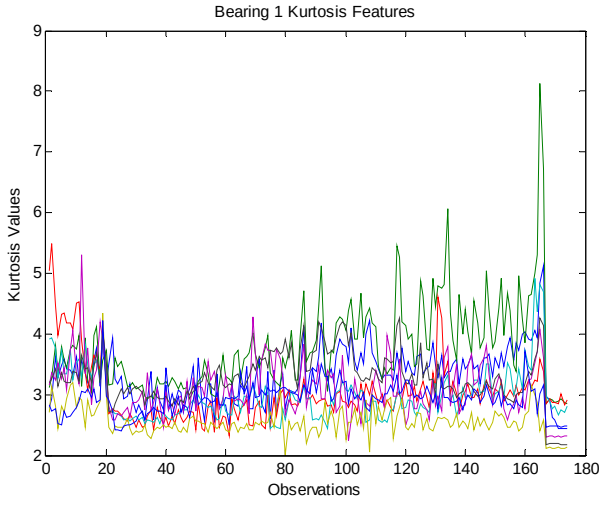


Figure 8.5-3: Bearing 1 Feature Plot: Kurtosis

Looking at the features in the plot, it can be clearly seen that some of the signals show a marked increase in kurtosis magnitude as the testing progressed. Features like these are useful for modeling because the change in the kurtosis indicates that the system is nearing failure, which

occurs here around observation 170. These features would need to be compared with the remaining bearing files to determine if a similar type of trend is present. So far, 4 other bearing data sets show this type of trend, while the others do not. As a last example, the RMS values over time for bearing 1 are shown in Figure 8.5-4.

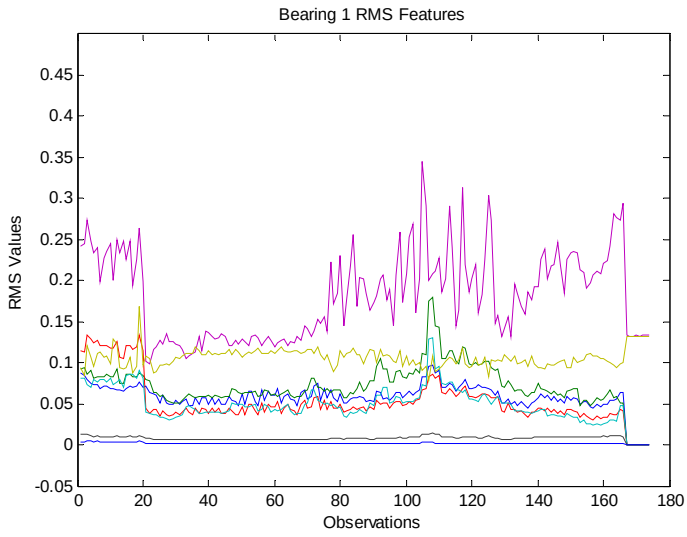


Figure 8.5-4: Bearing 1 Feature Plot: RMS

The RMS values for most of the signals in the plot show a change at observation 20 and then increase in value as testing progressed, with Signal 5 showing the largest amount of change. While not as clear a trend as seen in the kurtosis feature, the RMS feature also changes in magnitude along with the kurtosis feature such that both can be included when developing a prognostic parameter, though some additional processing to smooth the features will be needed. In total four additional bearing files show this type of trend in the RMS feature. In the next section, some preliminary results from analyzing the bearing data in the frequency domain will be shown.

Frequency Domain Preliminary Results

Another method that has been found useful for extracting feature information from raw signals is to examine the data in the frequency domain. As in the motor degradation data analysis, certain frequency peaks and frequency bands can be tracked through time or as testing progresses, and the results can be examined for trends. This method is typically accomplished by calculating the Fast Fourier Transform (FFT) and tracking the peaks through each FFT of each test file for all the bearings. Due to the nature of bearing operation, there will be certain frequency peaks from the

inner and outer race elements of the bearing, as well as a general ball pass frequency of the bearing. The power in these frequency peaks or the RMS values in these frequency bands can be tracked through each test, and thus the large amount of data can be reduced to a smaller and more usable size. Also, since the experiment involves rotation, other frequency peaks that can be tracked will be present in the FFT. As the bearing experiences more damage, the rotating bearing mount will experience those changes in the power of its dominant peak, and these changes can be tracked over time. To date, the exact model of the bearing is needed to determine the race and ball pass frequencies, so the preliminary frequency domain analysis has focused on looking for large changes in dominant peaks from the start of testing to the end of testing.

Figure 8.5-5 shows various snapshots of results from ultrasonic data collected on a failed test bearing. Each snapshot progresses in time (i.e. Figure 8.5-5 (a) being the earliest and Figure 8.5-5 (d) being the latest) and provides the Power Spectral Density (PSD) as a function of frequency for the ultrasonic sensor data. As illustrated in the figure, the amount of ultrasonic energy increases over the life of the bearing until failure occurs shortly after the snapshot shown in Figure 8.5-5 (d).

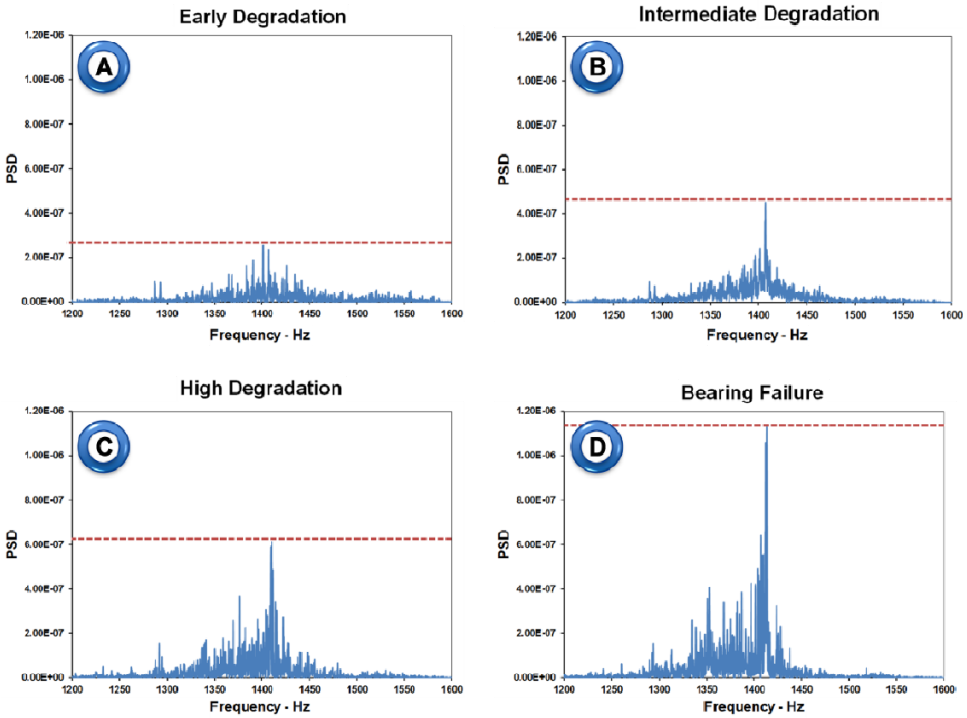


Figure 8.5-5 Progression of Ultrasonic Spectra through Time

Work in the frequency domain has focused on examining the change in size or power of the FFT from the first and last tests for each bearing. The dominant frequencies for each signal can be

identified, and later their maximum value can be tracked over testing. An example for Signal 1 of bearing 1 is shown in Figure 8.5-6. The top plot shows the FFT for the first test and the bottom plot shows the FFT for the last test.

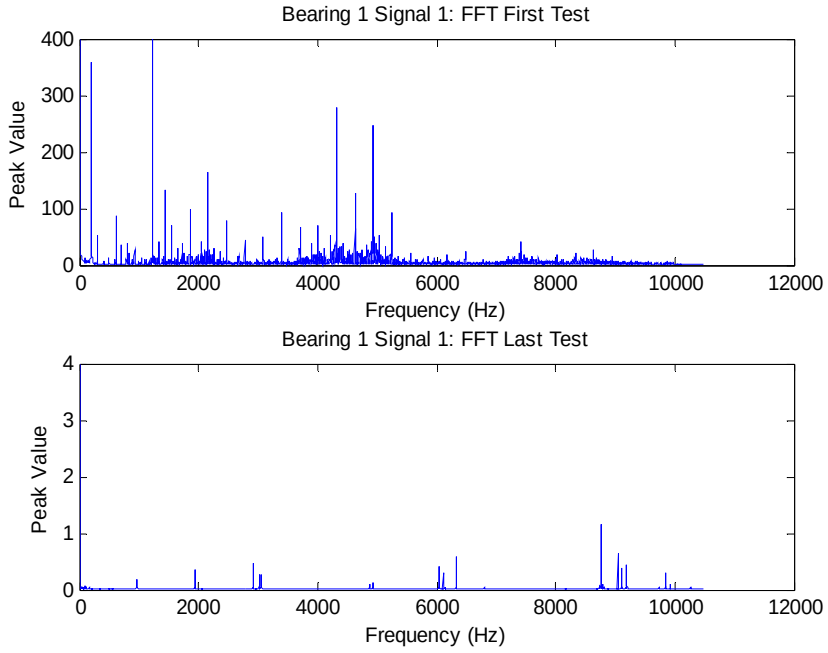


Figure 8.5-6: Bearing 1 Signal 1 FFT Plot

The major thing to notice in the FFT plot for this signal is the large reduction in the peak value from the first to the last test, which is on the order of 100 for this signal. The scale was set differently in the figure because the peaks in the bottom plot were impossible to see if using the same scale as the top plot. The preliminary results were found to be the same for each of the signals examined for several bearing files. Any of the large peaks in the top plot of the figure can be tracked through all tests, since all of the large frequencies significantly decreased in peak value as degradation increased. The first large frequency peak in Signal 1 of bearing 1 occurs at 200 Hz, so the maximum value of this peak in the range of 198-202 Hz is tracked through all 87 tests and is shown in Figure 8.5-7

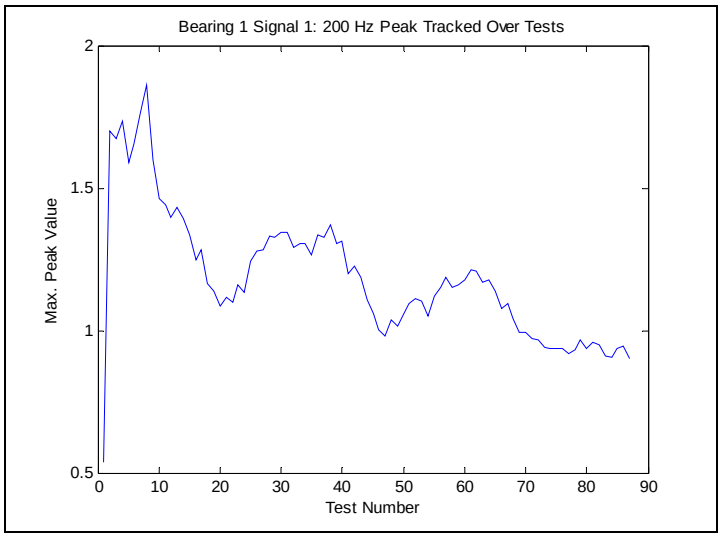


Figure 8.5-7: Bearing 1 Signal 1 Feature Plot: 200 Hz Peak Max Value Over Tests

Looking at the feature extracted from the 200 Hz peak of Signal 1 it can be seen that the value does indeed decrease over the course of testing so there is some assurance that the FFT plots show accurate information. Since the value of the feature shows a strong decreasing trend over time, this feature might be useful in further model development.

8.6 TASK 8: Degradation and Failure Data Collection

In this section we discuss the collection of degradation and failure data for the neoprene impeller test bed, and perform a preliminary analysis on the first sets of failure data. The impellers were heated to 165 C in an air circulated oven for approximately 1.5 hours. This reduction in the aging temperature from that reported previously of 190 C was necessary so that the pumps could be run for longer periods of time, at an aging temperature of 190 C the impellers failed within several hours up to 2 days. At the new aging temperature, the impellers typically fail within 3-5 days. Again, failure in this experiment means that the pump is no longer able to draw any water from the common sump. Finally, before any aging of the impellers was performed, new, unheated impellers were placed in each of the four pumps in order to collect baseline data. This baseline data can then be used in the future for comparison and model development.

For the initial data analysis, we will examine one of the pump data sets that show a clear indication of failure. The data is first cleaned using a median filter during discrete observation windows in order to remove any outliers and to de-noise the data. The Root Mean Square (RMS) values for each signal are first examined, followed by examination of the Power Spectral Densities (PSD). Last, a Joint Time Frequency Spectrum (JTFS) will be shown for each of the vibration, differential pressure and current signals for this test. The total observations recorded for

this particular test were 1107968 and several regions of interest are seen in the raw data that correspond to where an impeller broke off during testing. Table 8-6 lists these regions of interest.

Table 8-6 Degradation Regions for Accelerated Impeller Testing

Degradation Region	Observation Range
1	1 : 6.529 e4
2	6.530 e4 : 1.463 e5
3	1.465 e5 : 4.054 e5
4	4.056 e5 : 5.169 e5
5	5.172 e5 : end

In Table 8-6**Error! Reference source not found.**, regions 1-4 correspond to increasing degradation of the impeller with the pump still operational during these periods. In region 5, all of the impeller blades had broken off from the main impeller body and the pump was unable to draw any water from the common sump, which constitutes a failure. Next, Figure 8.6-1 shows the cleaned vibration, differential pressure and current signals for this test.

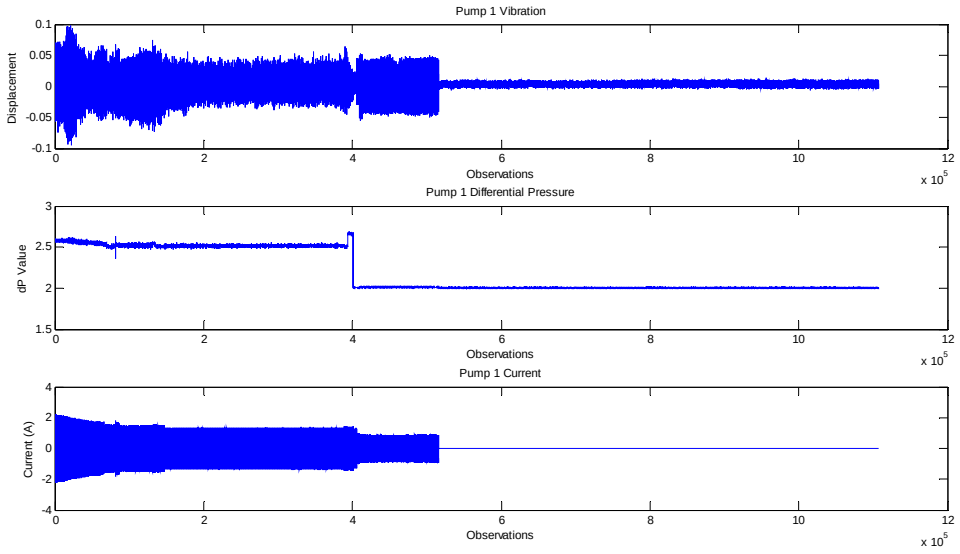


Figure 8.6-1: Cleaned Data for Various Signals of Pump 1

In Figure 8.6-1, the decreasing magnitudes of each of the signals correspond to one or more of the impellers breaking off until failure, which corresponds to the observation range of 5.2 e5 until the end of the recorded data. Next, Figure 8.6-2 shows the RMS values for each of these signals. These RMS values were calculated by using 541 windows and each window had 2048 data observations.

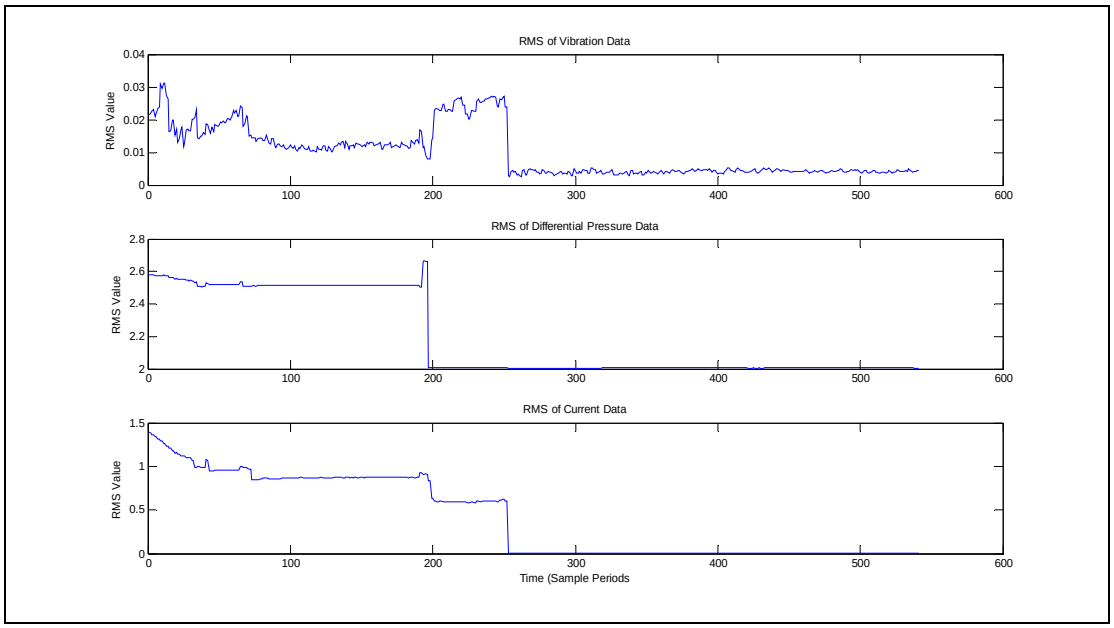


Figure 8.6-2 RMS Values for Various Signals of Pump 1

The degradation regions listed in Table 8-6 can also be clearly seen in the RMS values of the current data plot shown in Figure 8.6-2. Next, the PSD plots for the first four degradation regions of the vibration signal are shown in Figure 8.6-3.

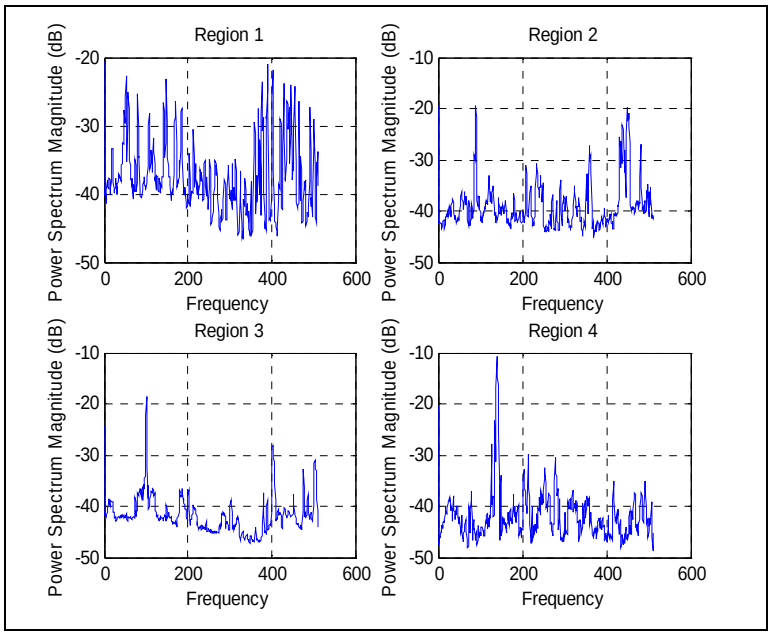


Figure 8.6-3 PSD Plots for Degradation Regions of Vibration Signal

The PSD for the first region is quite noisy; this was seen in all the pump data for this particular test. In region 2 there is a distinct peak around 120 Hz and at 420 Hz. The 120 Hz peak is seen in regions 2 & 3, while the 420 Hz peak disappears. Next, the PSD plots for each of the degradation regions for the current signal are shown in Figure 8.6-4.

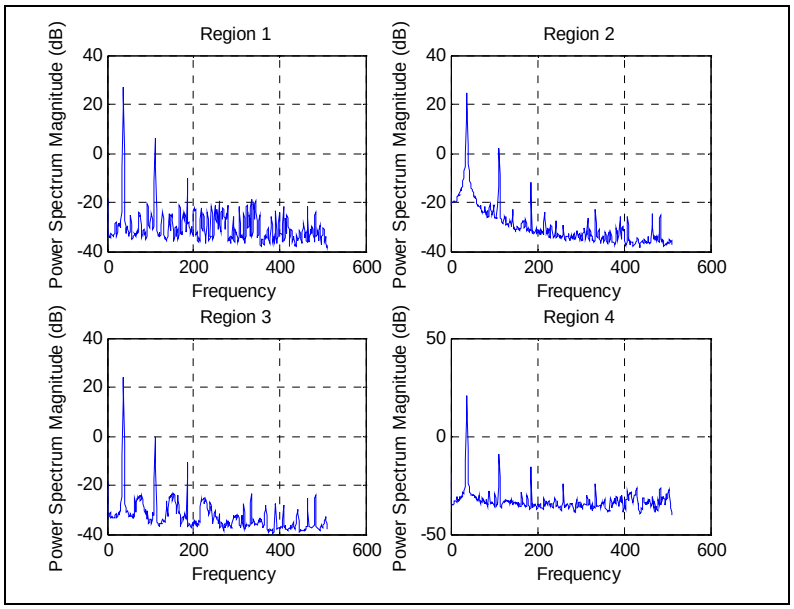


Figure 8.6-4: PSD Plots for Degradation Regions of Current Signal

In Figure 8.6-4, there are two noticeable peaks near 60 and 120 Hz and the magnitudes of the 120 Hz peak decreases during each region due to the fact that the pump is drawing less current as more impellers break off. The last PSD plots shown are for the differential pressure signal, which is shown in Figure 8.6-5.

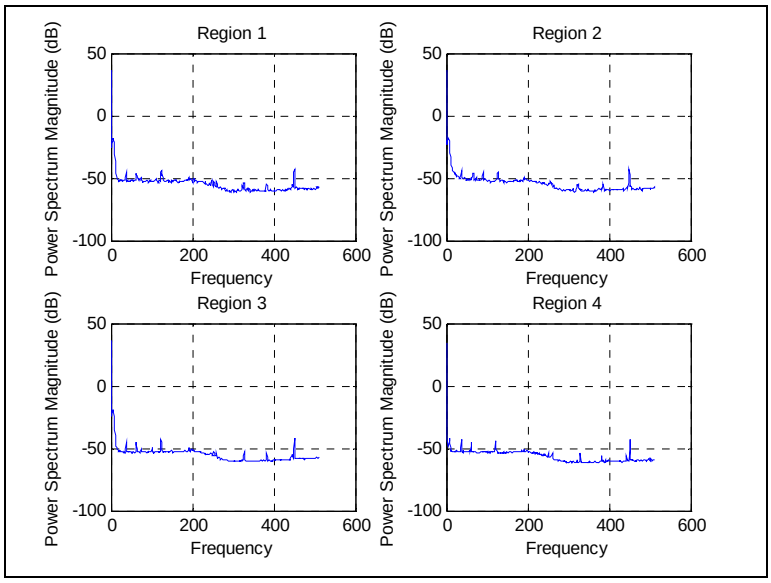


Figure 8.6-5: PSD Plots for Degradation Regions of Differential Pressure Signal

The PSD plots for the differential pressure don't show much change during each region. This is expected since this type of signal is not frequency based as the vibration or current signals are. Finally, JTFS plots are shown for the vibration and current signals. The differential pressure was not included in the JTFS plot because this signal did not contain any useful visible information. The JTFS plots contain the first four of the degradation regions as the final region contains little useful information since the pump has failed and is not drawing any water from the sump. Figure 8.6-6 shows the JTFS plot for the vibration signal.

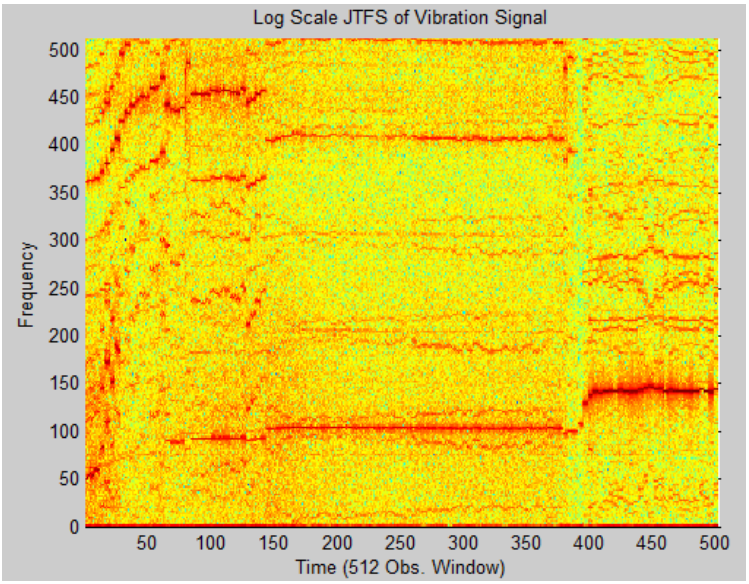


Figure 8.6-6: JTFS Plot for Vibration Signal

In Figure 8.6-6, the 50 Hz frequency grows as the test continues and shows a noticeable frequency at 120 Hz. The frequency at 360 Hz also appears to grow in magnitude to about 450 Hz until observation window 150, where it then lowers in magnitude to about 400 Hz. The last JTFS plot is shown for the current signal in Figure 8.6-7.

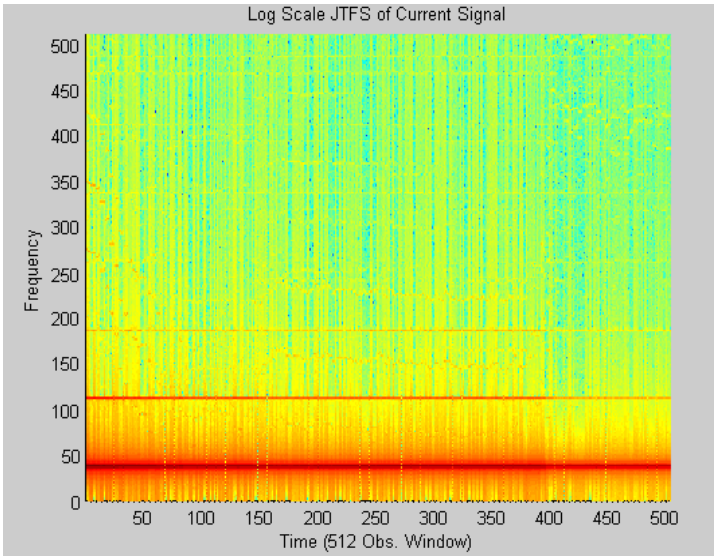


Figure 8.6-7: JTFS Plot for Current Signal

In Figure 8.6-7, there is a dominant frequency at 50 Hz that stays constant in magnitude throughout the test; this was seen in the PSD plots for this signal as well. Finally, the 120 Hz

frequency decreases in density as the test continues, which indicates that this frequency component is not as strong when all the impellers break off.

Summary and Conclusions:

In summary, the thermal aging temperature for the neoprene impellers has been reduced from 190 C to 165 C to ensure that adequate amounts of data can be collected. It was found that aging the impellers at 190 C caused the impellers to become too stiff and result in tests that last for only several hours to a day. Using a temperature of 165 C, it has been found that the testing can be performed continuously for 3-5 days. All four of the pumps on the test bed were used to test the impellers. Data analysis was performed on the data collected from one of the pumps and examination of the raw data for each of the monitored signals shows a clear indication of when an impeller blade broke off. The RMS values for each of the signals were also calculated and show indications of change in magnitude during each of the listed degradation regions. The PSD plots for the vibration and current signals show a change in the magnitude of certain peaks as the testing progressed, which means that the degradation can be tracked through time. Finally, JTFS plots for the vibration and current signals shows frequency components, namely 50-60, 120 and 360 Hz, show changes in density as the testing progresses.

9. Task 9: Use Algorithms to Develop Lifecycle Models

Any tool or algorithm, no matter how well thought out and mathematically based, should ultimately be tested on actual systems data in order to evaluate is usefulness and practicality. Data collected from both the on-site University of Tennessee experimental test beds and from the various test beds from the off-site project were used to verify the developed tools and algorithms of this project. The data analysis focuses on both the algorithms and tools developed during this project, and techniques previously established by industry and academia to be useful in the extraction of diagnostic and prognostic information. These additional techniques and analysis will serve as a benchmark allowing for a quantitative comparison with any tools developed in this project and ensuring that all technologies are state of the art. Independent analyses of the data, as well as models developed directly with tools developed in this project are used to highlight the usefulness and effectiveness of the developed algorithms. These results are presented in this section.

9.1 Analysis of Accelerated Motor Aging Data

This task describes the steps taken in analyzing the degraded motor data in the time and frequency domain. The data processing involves extracting useful features from the raw signals and using these features to develop prognostic models. Selecting appropriate features is an important factor when developing prognostic parameters, models and RUL estimations. Gathered signals from machine components generally contain large samples of data which require a large amount of memory and computation time to be analyzed. Instead, this data can be reduced into a lower but informative representation by extracting meaningful features from raw signals. The following section briefly describes the motor degradation experiment and data collection, as well as the monitored signals. Sections 9.1.2 and 9.1.3 describe the steps used to extract features from the data in the time and frequency domain. Section 9.1.4 describes the methods used to develop the prognostic parameter by use of the genetic algorithm found in the PEP toolbox and by Ordinary Least Squares (OLS) regression. Also included in this section is a reporting of the prognostic model results developed by using the extracted time and frequency domain features.

9.1.1 Experiment & Data Description

The data used during this research was steady-state data from a motor degradation project funded by EPRI. In the project, a group of ten 5 HP 3-phase motors were run through a degradation cycle on a weekly basis. First, the motors were heated for 3 days in an oven. The motors were separated into two different groups for heating degradation. One group, motors 1-7, was heated at 160

degrees C, while the other group, motors 8-10, was heated at 140 degrees C. After the first heating cycle, the motors were placed in a moisture testing bed with high humidity for further degradation. Then the motors were allowed to cool for a few hours before being placed in the second heating cycle for 3 additional days.

After the second heating cycle, the motors were placed on a testing bed and run for one hour. Five two-second transient startup tests were taken, including four loaded tests and one unloaded test. The steady-state data was collected every 15 minutes for one hour per motor. The following thirteen variables were monitored during this test at a sampling rate of 10240 Hz:

- Motor Current (3 phases)
- Motor Voltage (3 phases)
- X & Y Direction Accelerometer
- Microphone
- Tachometer
- Temperature
- Output Current (Generator)
- Output Voltage (Generator)

The data used in this research study was from five of the motors, which are numbers 2, 3, 5, 6 and 7, because these motors all present similar degradation due to bearings failures. Once the new set of motors being degraded shows evidence of failures then the same process will be applied to those motor data sets for model comparison. Next we will discuss how to extract features from the time and frequency domains.

9.1.2 Feature Extraction: Time Domain

Feature extraction from the time domain data is shown in this section, and the usefulness of the extracted features is examined. A usable feature would be one that shows a definite trend over the course of testing. This trend can be the magnitude of the feature value either increasing or decreasing over time. Less useful features are those that do not shown a trend over time. Several useful features have been extracted using statistical moments applied to the considered data and examining any trend over the entire lifetime of the tests for each motor. The most common statistical moments used in practice are the mean, standard deviation, root mean square (RMS), skewness and kurtosis. The mean value simply represents the average value of an observation

range, and it is obtained by dividing the sum of observed values by the number of observations, as shown in Equation 9-1:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \textbf{Equation 9-1}$$

The standard deviation gives an idea of how close the entire set of data is to the average value and is given in Equation 9-2 as:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_1 - \bar{x}\right)^2} \qquad \textbf{Equation 9-2}$$

where $\bar{x}$ is defined as the mean value.

The RMS, also known as the quadratic mean, is a statistical measure of the magnitude of a varying quantity. The RMS value of a set of values is the square root of the arithmetic mean of the squares of the original values. The skewness is a measure of the asymmetry of the data around the sample mean. If the skewness value is negative, the data are spread out more to the left of the mean than to the right. If the skewness is positive, the data are spread out more to the right. The skewness is defined in Equation 9-3:

$$SK = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^3}{\sigma^3} \qquad \textbf{Equation 9-3}$$

where $\bar{x}$ is defined as the mean value and σ is defined as the standard deviation.

Kurtosis is a measure of the "peakedness" of a distribution, often termed the fourth statistical moment, and it is defined by Equation 9-4 as:

$$KU = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^4}{\sigma^4} \qquad \textbf{Equation 9-4}$$

where $\bar{x}$ is defined as the mean value and σ is defined as the standard deviation.

These moments were used to analyze the steady state motor data and one value of each moment was calculated for each motor test. For example, motor 2 had 108 total tests so the features of

motor 2 all contain 108 data points. In this way the size of the data is reduced and hidden trends in the data over the lifetime of the motor can be observed. It is important to note that the trend has to be well defined, such as an increase in the feature value over time, and similar trends must be present in each motor to be considered useful. Some statistical features did not present a clear trend over time and they have not been considered as useful features to generate the prognostic parameters. If the trend over tests is not clear, the prognostic parameters generated will not have similar trends and the GPM model will give poor RUL estimation. Table 9-1 lists the time series features considered during the course of this research that showed a trend over time.

Table 9-1 Time Series Statistical Features

1	RMS Current 1	9	Kurtosis Current 3
2	RMS Current 2	10	Kurtosis Voltage 1
3	RMS Current 3	11	Kurtosis Voltage 2
4	RMS Voltage 1	12	Kurtosis Voltage 3
5	RMS Voltage 2	13	Kurtosis Current Output
6	RMS Voltage 3	14	Kurtosis Voltage Output
7	Kurtosis Current 1	15	RMS Vibration-X
8	Kurtosis Current 2	16	RMS Vibration-Y

Looking at Table 9-1, it can be seen that the majority of the signals that showed any real trend over the course of testing were the current and voltage signals. This result makes sense because the degradation testing causes shorts and surges in these signals and thus would show in the signals as something wrong with the system. The vibration signals only had two usable features in the time domain but as will be seen later, the vibration signals had more useful information in the frequency domain. An example of a set of features that is not considered usable is shown in Figure 9.1-1, which is the mean of the current signals over time for Motor 2.

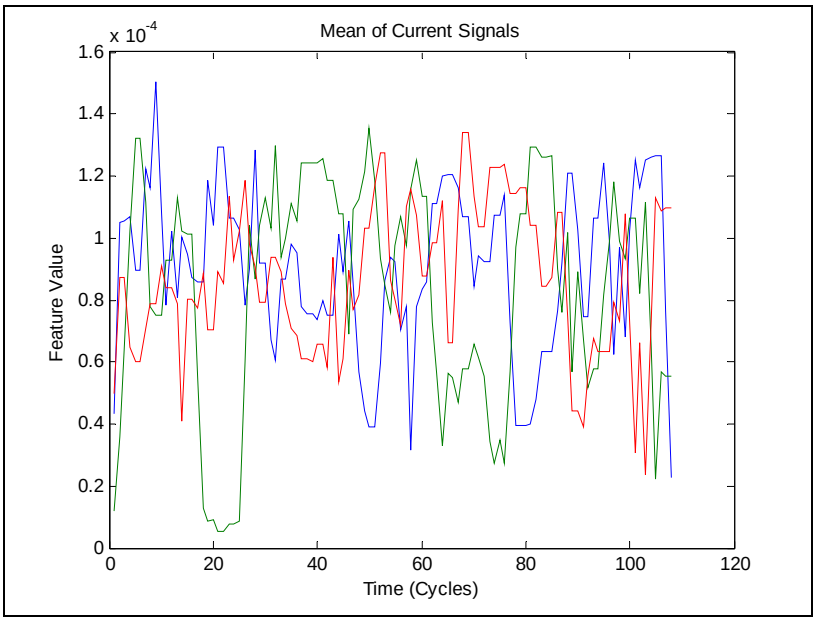


Figure 9.1-1: Unusable Features from Current Signals

The features shown in the figure are unusable due to the fact that there is no clear increasing/decreasing trend. The plots can in general be considered constant over time. Constant valued features provided no useful information and the prognostic parameters generated from such features would not give reliable estimates of RUL. Another example of a feature that is not useful for prognostic parameter generation is shown in the figure below, which shows the skewness values of the voltage signals in Motor 2 over time.

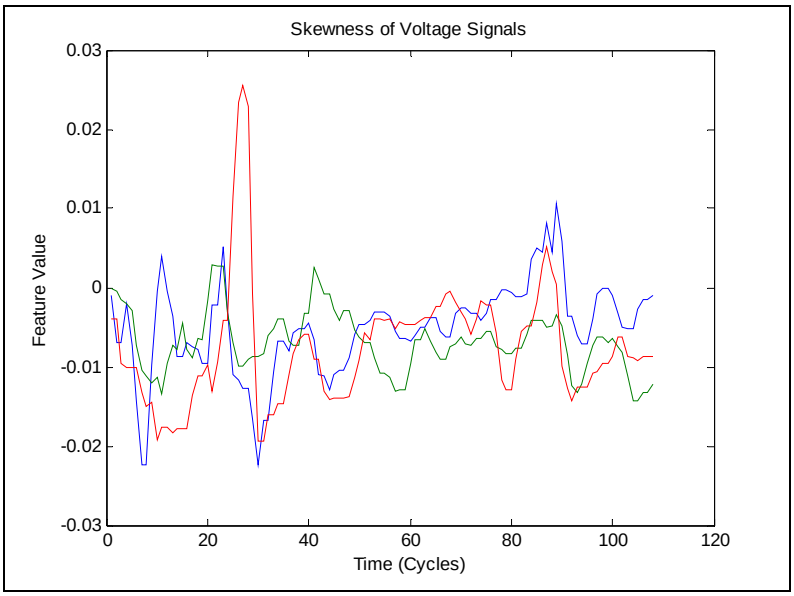


Figure 9.1-2: Unusable Features from Voltage Signals

Besides the large spike in the red plot shown in the figure, there is again no observable trend in the data; the skewness values are generally constant over the course of testing. An example of a usable feature is shown in Figure 9.1-3, the RMS of the voltage signals in Motor 2 over the course of testing.

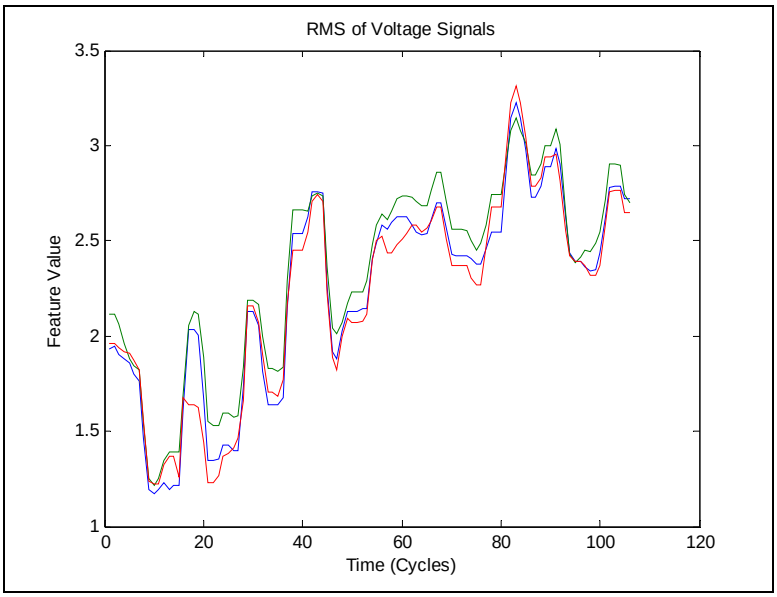


Figure 9.1-3: Usable Features, RMS Values for Voltage Signals

The bearings in the figure all show a general increase in value over time and were considered a usable set of features. Another example of a useful feature is shown in Figure 9.1-4, which are the kurtosis values of the voltage signals in Motor 2 over time.

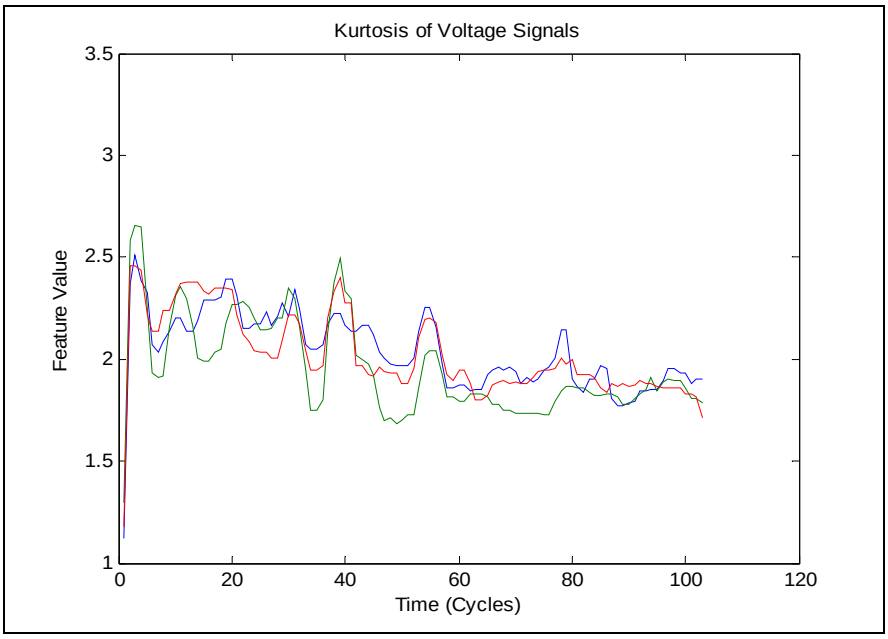


Figure 9.1-4: Usable Features, Kurtosis Values for Voltage Signals

In this case, the values shown in the figure all show a decreasing trend over the course of testing, and thus kurtosis was included as a useful feature. To remove large spikes in the data and features in order to clarify the underlying trend, a band reject filter that eliminates the 60 Hz component has been applied to the original signals. This filter was applied because the 60 Hz component can dominate the behavior of the sampled data and hide trends in the data. The filtering process involves taking the FFT of the signal, removing the data corresponding to the 60 Hz values and then inverting the FFT to obtain the time series signal. Care must be taken because the inversion of the FFT can introduce anomalies at the beginning and end of the de-noised signal and should not be considered as trends or a feature in the data. In Figure 9.1-5 the RMS values for the voltage signals of Motor 5 are shown, without the band pass filter applied.

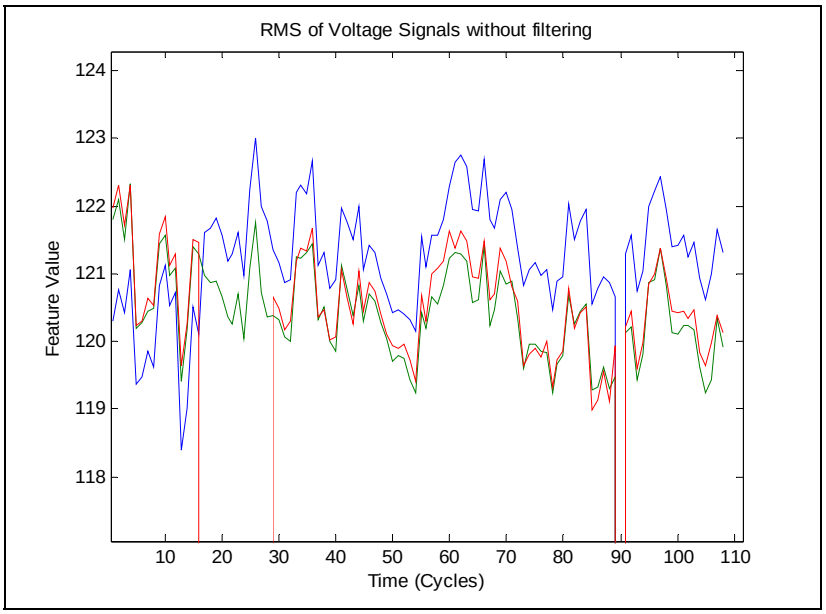


Figure 9.1-5: RMS of Voltage Signals, No Band-Pass Filter

It can be seen in the figure that there is no clear trend over time and the RMS is likely not a usable feature. However, applying the band pass filter to the signals and taking the RMS values results in an observable trend, which is shown in Figure 9.1-6.

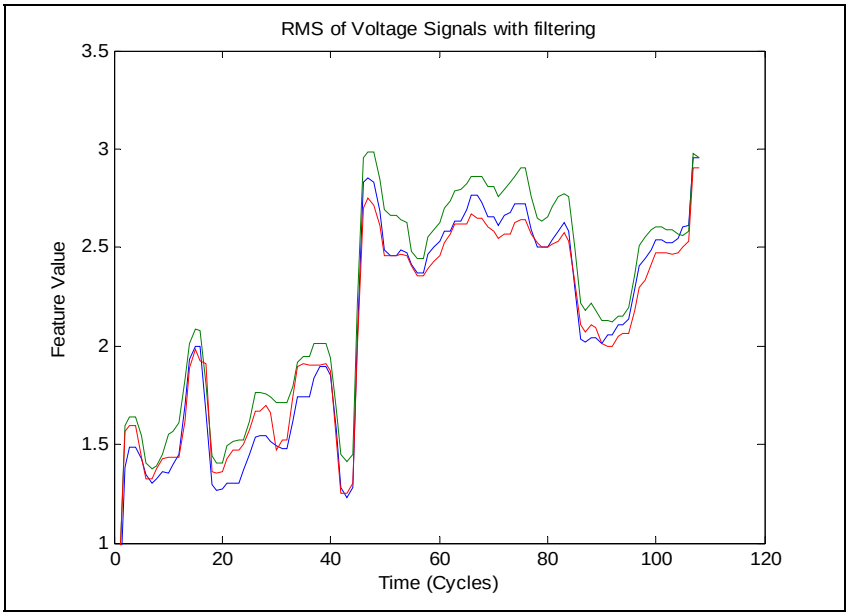


Figure 9.1-6: RMS of Voltage Signals, Band Pass Filter Applied

The same band pass filter was applied to the motors of interest and each of the time series features were generated by using this filtering method. As shown in Table 1, 16 usable features were extracted from the motor data sets. These features can be used alone or in combination with the frequency domain generated features. Incorporation of these features into the prognostic model will be shown later. Next the methods used to extract useful features in the frequency domain are discussed.

9.1.3 Feature Extraction: Frequency Domain

Generation of useful features in the frequency domain is described in this section. Frequency domain analysis is based on transforming the time series signal into the frequency domain. The main advantage of frequency domain analysis over time domain analysis is its ability to identify and isolate certain frequency components of interest. Features extracted from the frequency domain can generally indicate machinery faults better than time domain features, especially in the case of vibration signals, because characteristic frequency components such as resonance frequency components or defect frequency components can be relatively easily detected and matched to faults [McInerny & Dai]. Frequency domain or spectral analysis of the vibration signals is perhaps the most widely used approach to bearing defect detection. The modern Fast Fourier Transform (FFT), **Equation 9-5**, is the most conventional diagnosis technique and has been widely used to identify the frequency features of signals.

The FFT for a discrete data series is defined as:

$$X(k) = \sum_{i=1}^{n} x_i \omega_N^{(i-1)(k-1)} \qquad \textbf{Equation 9-5}$$

where $\omega_N = e^{(-2\pi)/N}$ is an N[th] root of unity.

In this research, the FFT has been applied to extract features from the two vibration signals, which typically contain useful bearing failure information. Features have been obtained by measuring a certain peak value in the frequency spectrum for each motor and by calculating the RMS values in the frequency domain over a specific frequency band of the frequency spectrum. For example, the RMS values are calculated in the band containing bearing fault frequency information [Li et el]. The ball pass frequencies of the bearing are used for peak tracking in the frequency domain and occur at the frequencies listed below:

- Ball pass frequency of the inner race [325 Hz]
- Ball pass frequency of the outer race [215 Hz]

- General ball pass frequency [283 Hz]

The current and voltage signals did not contain any useful information in the frequency domain, and these signals were not considered during the frequency domain analysis. Figure 9.1-7 shows the features extracted from the X-direction vibration signal of Motor 7, which tracks the peak values of the aforementioned bearing fault frequencies.

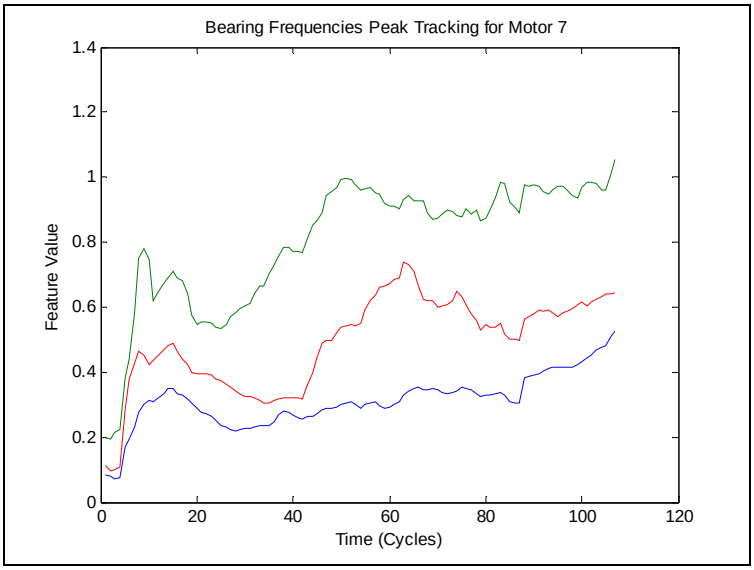


Figure 9.1-7: Peak Tracking of Bearing Fault Frequencies, X-Direction Vibration

The features in the figure show that as the testing progress, the energy contained in each of the bearing fault peaks increases. This clear increasing trend is considered a useful feature of the frequency domain data. Another example of the trend in the bearing fault frequencies is shown in Figure 9.1-8, which is the bearing peak values for the Y-direction vibration of Motor 2 tracked over testing.

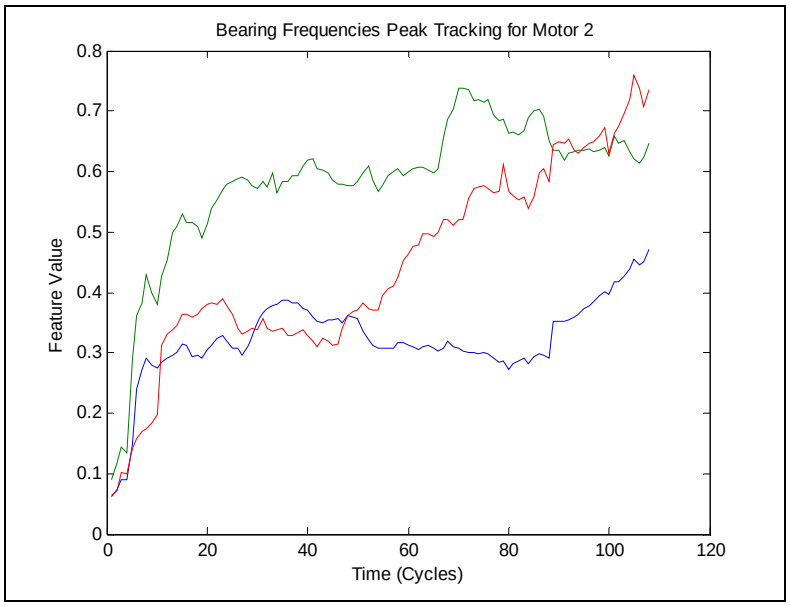


Figure 9.1-8: Peak Tracking of Bearing Fault Frequencies, Y-Direction Vibration

As shown in the figure, the peak values of the bearing fault frequencies all increase in value over time. To show that these features are the same basic shape for each motor tested; Figure 9.1-9 shows the peak tracked features for the Y-direction vibration for all other motors.

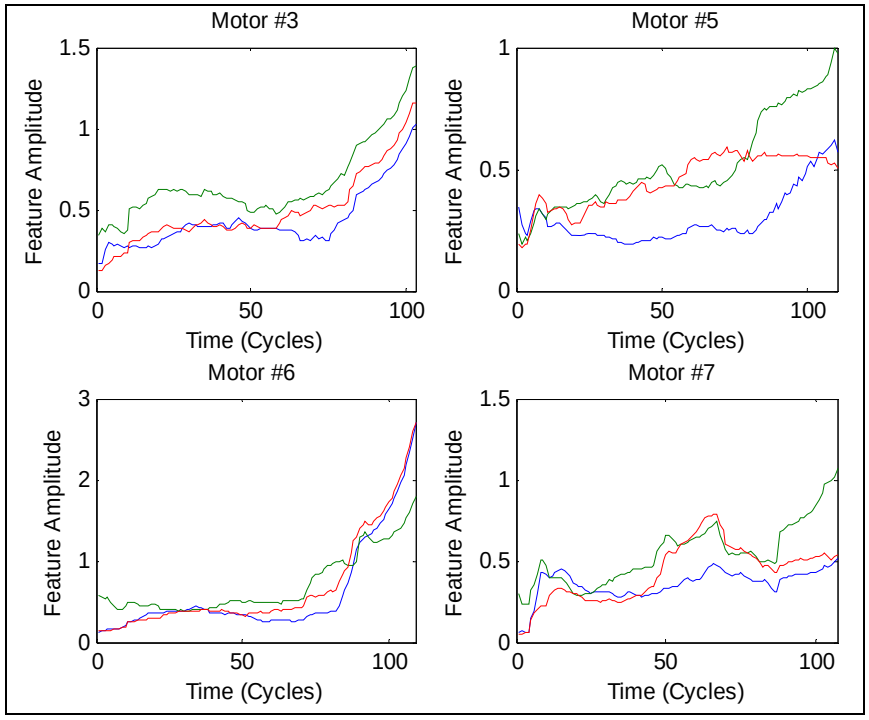


Figure 9.1-9: Peak Tracking, Y-Direction Vibration

It can be seen in the figure that except for Motor 7, all other features show an increasing value as testing progresses. Although Motor 7 did not have a strong change over time, it was still included in the analysis for completeness.

The next features that have been extracted from the frequency data involve taking the RMS values over a frequency range and tracking the value through all tests. As the three bearing faults show frequencies in the range of 215-325 Hz, the RMS values in the X & Y vibration FFT data for Motor 2 have been calculated for this range and are shown in Figure 9.1-10.

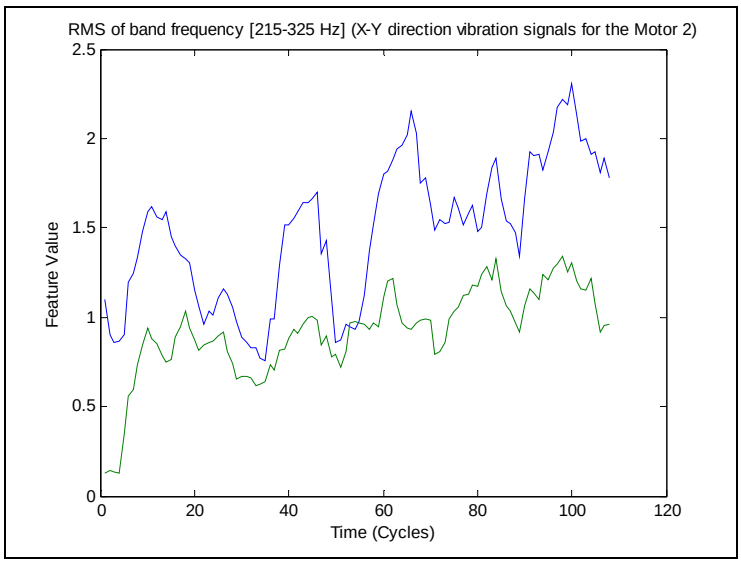


Figure 9.1-10: RMS Values of X & Y Vibration, Frequency Range of 215-325 Hz

The figure shows that the RMS values of the two vibration signals show an increasing trend over time, but the X-direction trend, shown in blue, is much more pronounced. This trend indicates that the feature might be useful for modeling and prognostic parameter generation. Next, the RMS values over the same frequency range for the other motors are calculated to see if the same trend appears. The results are shown in Figure 9.1-11.

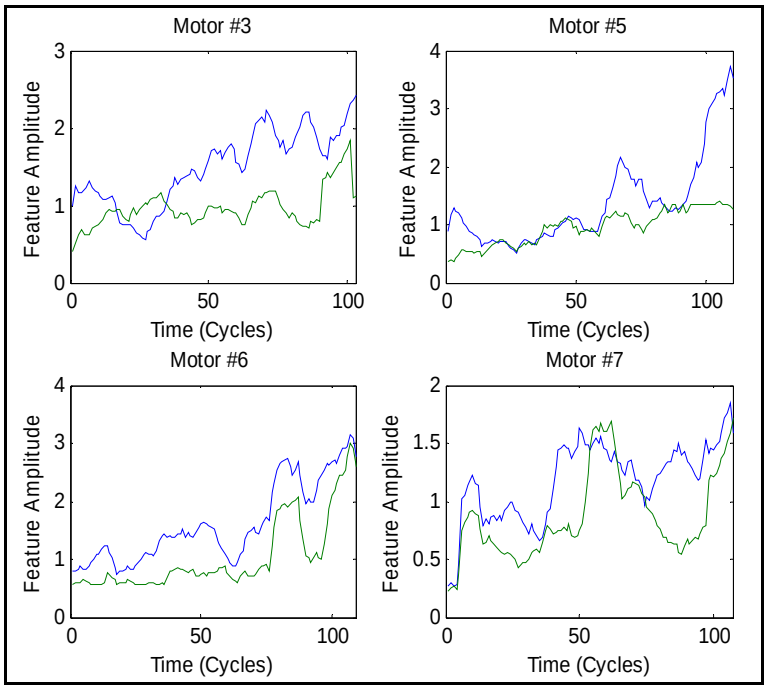


Figure 9.1-11: RMS of Frequency Band, Vibration Signals of Remaining Motors

Looking at the previous figure, it can be seen that the remaining motors all follow a trend in the RMS values calculated in the frequency band. The X-direction vibration feature, shown in blue, has the strongest trend for all motors, while the Y-direction, shown in green, is not as well defined, though it may still be useful for modeling. Hence, in addition to the 16 time domain features, an additional 8 features have been extracted from the vibration signals in the frequency domain. These new features are listed as follows:

- Features 17 to 19 were extracted by peak tracking ball pass frequencies, X-direction vibration
- Features 20 to 22 were extracted by peak tracking ball pass frequencies, Y-direction vibration
- Features 23 and 24 were extracted by taking RMS values over frequency values of bearing fault band [215-325 Hz]

To summarize this section, the steady state data collected during the motor degradation experiment was analyzed in the time and frequency domain to extract useful features for prognostic parameter and model development. The signals consisted of current, voltage, vibration, tachometer measurements of the motor, and the output current and voltage of the generator. Only the tachometer signal was found to be unusable during the analysis, while the

other signals had interesting features in the time and frequency domain. The majority of this analysis focused on extracting features from the raw signal. In the time domain, the RMS and kurtosis values for the current, voltage and vibration signals showed interesting trends over time and were considered for further analysis. The mean and skewness values for these and other signals were fairly constant in value as testing progressed and were not usable in the development of prognostic models.

In the frequency domain, the bearing inner and outer race frequency, as well as the general ball pass frequency, were tracked over time. These peaks occur at 325, 215, and 283 Hz, respectively. The simple peak tracking analysis involved storing the maximum power value of each of the bearing frequencies in a small band. For example, the inner race peak was tracked from 320-330 Hz and the maximum value in this small frequency range was stored. Both the vibration signals showed increasing trends in these values as testing progressed. Another useful feature extraction method involves taking the RMS values of the entire bearing frequency and tracking these values over time. For this analysis, the band from 215-325 Hz was used, and trends were seen in the features that were considered useful for further modeling. The current and voltage signals of the motor and generator did not show any usable feature trends when analyzed in the frequency domain, as the power in the frequency did not change dramatically.

9.1.4 Modeling and Prognostic Parameter Development

In this section, Genetic Algorithm (GA) and Ordinary Least Squares (OLS) estimation techniques were used to develop suitable prognostic parameters from the selected motor features. The generated prognostic parameters can then be used to develop the General Path Model (GPM) and obtain predictions of RUL. The GPM is constructed for the GA and OLS cases, and the "Prediction Metrics" function will be used to offer quantitative results for prognostic models developed using the parameters. The prognostic parameters, failure times, and developed models will be used in future work to obtain RUL predictions for the current motors being tested and to validate performance of the algorithms developed during this course of research.

Genetic Algorithm Approach for Prognostic Parameter Generation

The PEP toolbox's "optparam" function uses genetic algorithms that are contained in the MATLAB Global Optimization Toolbox® that theoretically generates a near-ideal prognostic parameter from the features shown in the previous section. The genetic algorithm is a unique optimization method that uses natural selection rules similar to those discovered in biology to arrive at an optimal solution to problems that may be constrained or unconstrained. The algorithm

randomly chooses "parents" from the data source to create "children" in successive iterations. This selection process is performed iteratively to potentially arrive at an optimal solution of the problem. More information on genetic algorithms can be found in the help sections in the MATLAB toolbox and in [Haupt & Haupt, 2004]. Recall that a prognostic parameter must ideally have three characteristics that are defined as monotonicity, prognosability and trendability [Coble 2010]. Monotonicity helps to characterize the positive or negative trend of the parameter; trendability helps to show if the parameter or parameters in the population can be fitted to a similar underlying functional form; finally, prognosability characterizes failure value spread in relation to the parameter pathway. Each parameter metric is assigned a value between 0 and 1, where an ideal or perfect parameter has a value close to or at 1.

The "optparam" function arrives at an optimized prognostic parameter by using a fitness function that sums the three parameter characteristics, with an optional weighting for each characteristic. The default in the function is to arrive at linear combinations of the three characteristics from various data sources using a fitness function of the form:

$$fitness = w_m monotonici\ ty + w_t trendabili\ ty + w_p prognosabi\ lity \qquad \textbf{Equation 9-6}$$

Each of the weights for the performance metrics in the preceding equation is initially set as [1 1 1] such that equal importance is given to each of the three metrics. Optparam uses this fitness function to find optimal weights for a linear combination of extracted features supplied to it. This optimally weighted linear combination of features can then be used as the prognostic parameter, indicative of the system degradation, and useful for creating Type III condition-based prognostic models.

For validation purposes during the model development, the "leave one out" method was used. This term means that the linear combination weighting values generated from the optparam function were calculated for 4 of the motors while leaving the data source for 1 of the motors out. The generated weights are then multiplied by the removed motor data to create prognostic predictions from the generated model. Ideally, the prognostic parameter of the left-out data source should have the same shape as the other four generated parameters.

The main problem that can be encountered using the genetic algorithm approach is that during the optimization process the genetic algorithm can become stuck in a local minimum point and not reach the global minimum point of the error surface. This means that the generated weights are not guaranteed to be absolutely optimal. Also, given the random starting points of the genetic

algorithm, each run of the program can result in different optimized weights from the same data sources. Finally, if using large data sources with many features, the genetic algorithm can be computationally intensive. In this research the genetic algorithm method was investigated but ultimately not used due to the fact that the OLS method explained in the next section offered better end results. Both methods are available for use in the diagnostic/prognostic toolbox developed. The same models were developed using both methods for comparison.

Ordinary Least Squares Approach for Prognostic Parameter Generation

Another method that was investigated to arrive at a near-ideal prognostic parameter is to use OLS estimation. In this method, "X" is defined to be a matrix of the features for each of the 5 motors that were investigated. Each column in the matrix is the same feature from each of the 5 motors, resulting in a 537 x 16 matrix. Hence, 537 data points were collectively extracted from each of the 5 motors, and 16 features were investigated. Matrix "Y" is defined as a time matrix that is scaled between 0 and 1 for each motor. The "leave one out" validation method is also employed here to validate that the resulting prognostic parameter of the left-out motor has the same general shape and metric values as the other 4 motors. Once the **X** and **Y** matrices are defined, the resulting weights can be found by:

$$w = pinv(X^T X)X^T Y \qquad \textbf{Equation 9-7}$$

Multiplying the resulting weights by **X**, the prognostic parameter is obtained. This method was preferred because the computation time is very short compared to genetic algorithms, and the prognostic models had better performance when using the OLS method. It is also noted that mean, median and exponential filters were also applied to these prognostic parameters to investigate if filtering had any effect on the final model results. As an example of what the resulting prognostic parameter looks like, Figure 9.1-12 shows the resulting parameter using the OLS method and using different combinations of features.

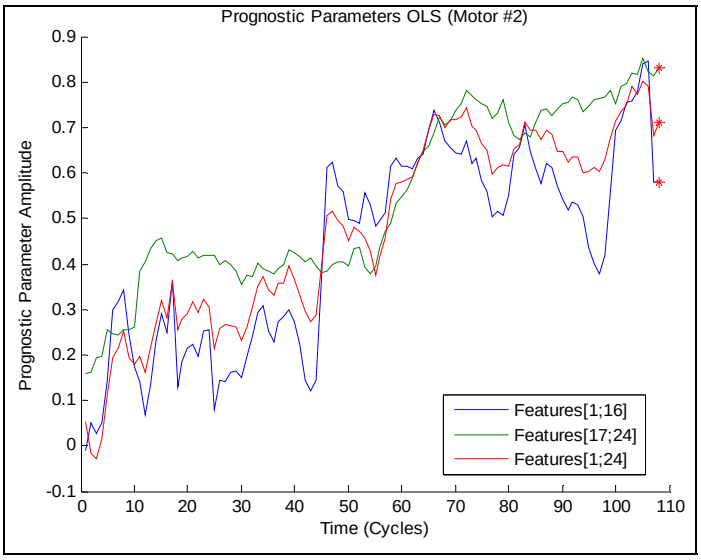


Figure 9.1-12: OLS Prognostic Parameters Using Different Features

Looking at the figure, it can be seen that the shape of the prognostic parameter changes depending on what features are used. The general trend of these parameters is linear in functional form so a linear GPM can be built with these parameters. Once the prognostic parameters are developed, the GPM could be constructed and RUL predictions made. As a further example, the results for the three prognostic metrics are shown in Table 9-2 for both the OLS and GA methods using several combinations of features.

Table 9-2 Prognostic Parameter Results for OLS & GA

MOTOR #2	**OLS**			**GA**		
Selected Features	**M**	**P**	**T**	**M**	**P**	**T**
[1;16]	0.83	0.84	0.88	0.90	0.91	0.85
[1;14]	0.83	0.87	0.84	0.90	0.97	0.82
[1;6]	0.67	0.83	0.65	0.90	0.86	0.68
[1;9]	0.83	0.88	0.84	0.83	0.98	0.82
[7;14]	0.83	0.93	0.61	0.90	0.89	0.79

[1 4 5 6 9 12]	0.67	0.81	0.64	0.83	0.89	0.71
[1 4 5 6 9 12 13 14]	0.67	0.91	0.80	0.83	0.91	0.71
[17;22]	0.67	0.76	0.55	0.83	0.80	0.79
[17;24]	0.67	0.72	0.65	0.83	0.98	0.59
[1;14 17;24]	0.83	0.85	0.85	0.90	0.99	0.85
[1;24]	0.83	0.85	0.86	0.91	0.99	0.89

Looking at the table, it can be seen that the GA results are slightly better than the OLS method. It should be noted that the GA results are not as constant as the OLS results. By the random nature of the GA process, the prognostics metrics will be slightly different each time the process is run. Next are shown the results for each of the prognostic models developed that use prognostic parameters generated from several different combinations of the 24 features. The GA and OLS results are compared to see which offers the best predictive model and performance.

9.1.5 Prognostic Model Results

In this section the prognostic model results when using the various 24 features extracted from the steady state motor data are shown. The prognostic parameters were developed from the GA and OLS method so comparisons could be made as to which method gave the best prognostic model performance. A linear GPM model was chosen for the current work as the resulting prognostic parameters followed this basic trend, though a quadratic, cubic or exponential model could also be developed if the parameters show those types of trends. After the GPM is run using PEP toolbox functions, the RUL and time predictions are saved so that comparisons of model results when using various combinations of features could be examined.

To compare the performance of the various prognostic models, the "Prediction Metrics" function [Sharp 2013] is used, which uses the RUL and time predictions and the actual Time of Failure (ToF) to quantify model performance. The function outputs several scores based on the inputs and are:

- Aggregate Score (%) – gives an overall performance of the prognostic model based on error, uncertainty and coverage metrics. A score of 100% is considered perfect prediction while a score under 50% is an undesirable model.
- MAE – gives the mean absolute error and standard deviation of all cases of supplied input. A small error and deviation is desired

- WEB (%) – gives the Weighted Error Bias in percentage, a measure of the bias, positive or negative, of the model predictions. A small bias percentage throughout prediction time is desired.
- WPS (%) – gives the Weighted Prediction Spread in percentage, a measure of how much the predictions are spread. A small percentage spread is required.
- 95% C.I. Coverage (%) – a measure of the 95% Confidence Interval Coverage, or the percent of time that the estimated prediction covers the true RUL. The metric should ideally be at 95% or above.
- 10 % RUL C.H. (%) – a measure of the percentage of RUL where prediction uncertainty covers the true RUL and is less than a specified tolerance level. A value close the 10% level is preferred.

All the metrics listed contribute to the aggregate score of the model, and the results are dependent on the user input. Features that are well chosen and defined will lead to prognostic parameters with higher metric values and thus better RUL predictions. The function also gives these metrics in a graphical format.

The first results shown are for prognostic models developed using parameters generated from various combinations of features. Table 9-3 shows the results of the models when all or any combination of features is used for the OLS method. Table 9-4 will show the prognostic model results when all or combinations of features are used for the GA method. The entries highlighted represent the models with the best performance.

Table 9-3 : Prognostic Model Results Using OLS Method

Ordinary Least Squares Estimation Approach						
Features selected	**Aggregate score**	**MAE**	**WEB [%]**	**WPS [%]**	**Confidence interval coverage [%]**	**Convergence Horizon [%]RUL**
[1;16]	68.76	4.99+/-4.10	-1.02	23.95	100	0
[1;14]	68.26	6.02+/-4.14	-1.78	25.17	100	0
[1;6]	68.13	5.51+/-2.16	-2.16	20.56	95.2	0
[1;9]	68.59	5.51+/-2.16	-0.90	24.75	100	0

[7;14]	66.79	11.37+/-2.76	-4.75	28.09	100	0
[1 4 5 6 9 12]	66.95	9.45+/-3.49	-3.39	28.79	100	0
[1 4 5 6 9 12 13 14]	65.12	10.44+/-5.99	-3.19	36.34	100	0
[17;22]	64.65	10.34+/-7.36	-4.43	36.98	100	0
[17;24]	67.35	8.29+/-2.96	2.19	28.40	100	0
[1;14 17;24]	70.58	3.71+/-0.78	1.63	16.06	100	0
[1;24]	71.06	3.49+/-1.78	-0.32	15.46	100	0

Looking at the table, it can be seen that the model that performed the best in the OLS used all of the 24 features extracted from the time and frequency domains. This model had the best overall aggregate score and the least amount of error in the RUL predictions. The other models also performed well in that the difference in the MAE values and aggregate scores was at most 6% different for both metrics. Next, the GA prognostic model results are shown in Table 9-4.

Table 9-4 Prognostic Model Results Using GA Method

						Genetic Algorithm Approach
Features selected	**Aggregate score**	**MAE**	**WEB [%]**	**WPS [%]**	**Confidence interval coverage [%]**	**Convergence Horizon [%]RUL**
[1;16]	61.79	21.65+/-9.13	-9.43	33.89	81	9.5
[1;14]	62.11	18.43+/-11.87	0.23	51.32	100	0
[1;6]	71.51	3.35+/-1.34	-1.19	12.78	90.5	9.5

[1;9]	56.39	22.78+/-12.27	6.09	68.36	100	0
[7;14]	55.86	30.32+/-22.41	-2.44	64.59	90.5	0
[1 4 5 6 9 12]	61.45	16.45+/-9.14	-4.26	35.36	85.7	0
[1 4 5 6 9 12 13 14]	57.22	23.97+/-6.61	-2.49	54.35	85.7	0
[17;22]	64.01	11.03+/-6.28	1.42	42.56	100	0
[17;24]	65.95	9.96+/-5.81	1.35	34.87	100	0
[1;14 17;24]	65.43	10.04+/-4.09	-4.62	24.12	90.5	0
[1;24]	68.03	9.02+/-3.46	-1.95	25.91	100	0

Looking at Table 9-4, it can be seen that the model that best performed used only the first six features from the time domain, which were the RMS values of the motor current and voltage signals. This model is also one of the few that were able to achieve a 10% RUL convergence horizon, while none of the OLS models reached this convergence. It is also interesting to note that the GA models had much higher MAE than the OLS models. The reason for this is again the random nature of the GA when developing the prognostic parameters. The parameters using the GA method were also noisier than the OLS parameters. Next the Performance Metrics function graphical output is examined for the best models for the OLS and GA cases. Figure 9.1-13 shows the output for the OLS case when all 24 features are used for RUL predictions.

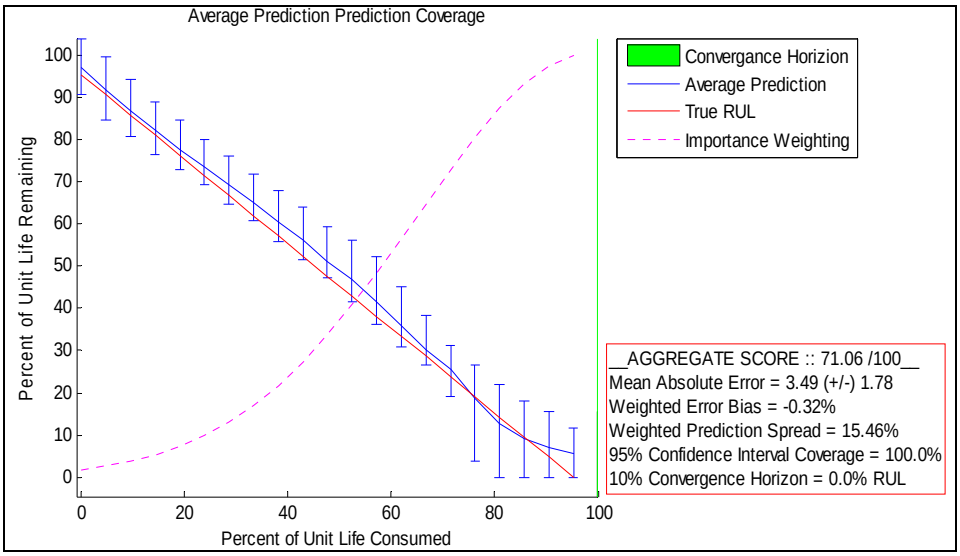


Figure 9.1-13: Performance Metric Output for OLS Case of All 24 Features

Looking at the figure, the blue line, which is the average RUL prediction, follows closely to the true RUL, shown using a red line. There is a slight deviation near the end of life, but this difference is not so large that RUL predictions cannot be made. The only problem with this model is that it did not reach the 10% convergence horizon, which is the green bar on the y-axis. To correct this issue, the prognostic parameters could be run through smoothing algorithms to change these results. Next is the GA model plot using all 24 features for comparison in Figure 9.1-14.

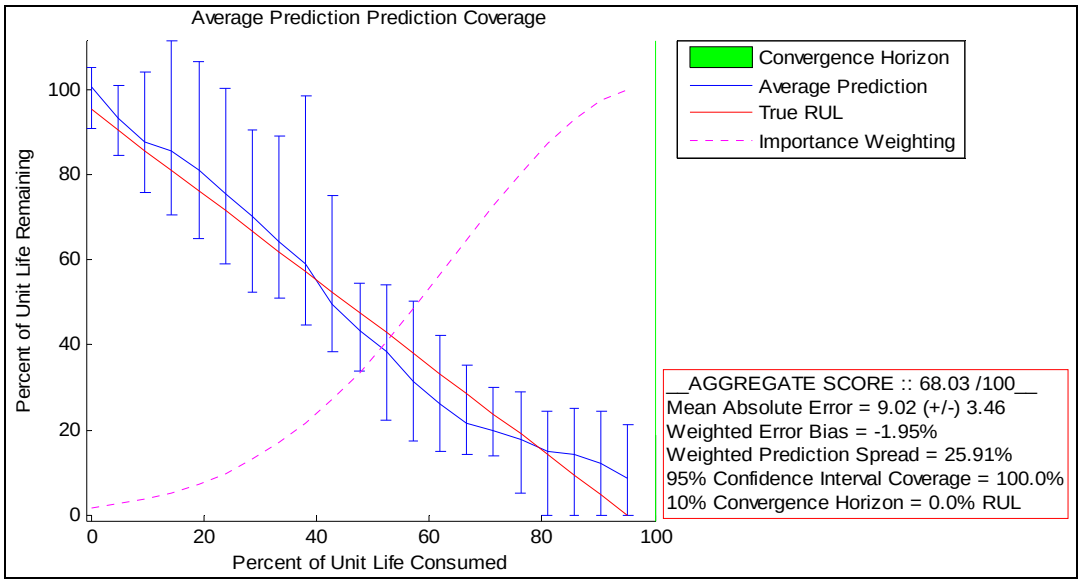


Figure 9.1-14: Performance Metric Plot for GA Method Using All Features

Looking at the figure for the GA results, the confidence intervals for the average RUL predictions seem very large from 0-40% of life consumed. This effect is due to the noisier prognostic parameters generated by the GA and is typically not seen as a problem because the large deviation in the RUL predictions occurs at the start of life and not at the end of life, which is normal. This model also had a lower aggregate score than the OLS model and had a MAE almost 7% larger, and this model did not reach the 10% RUL convergence horizon. The best GA model used only the first six features in the time domain, which were the RMS values of the current and voltage signals. Figure 9.1-15 shows these results.

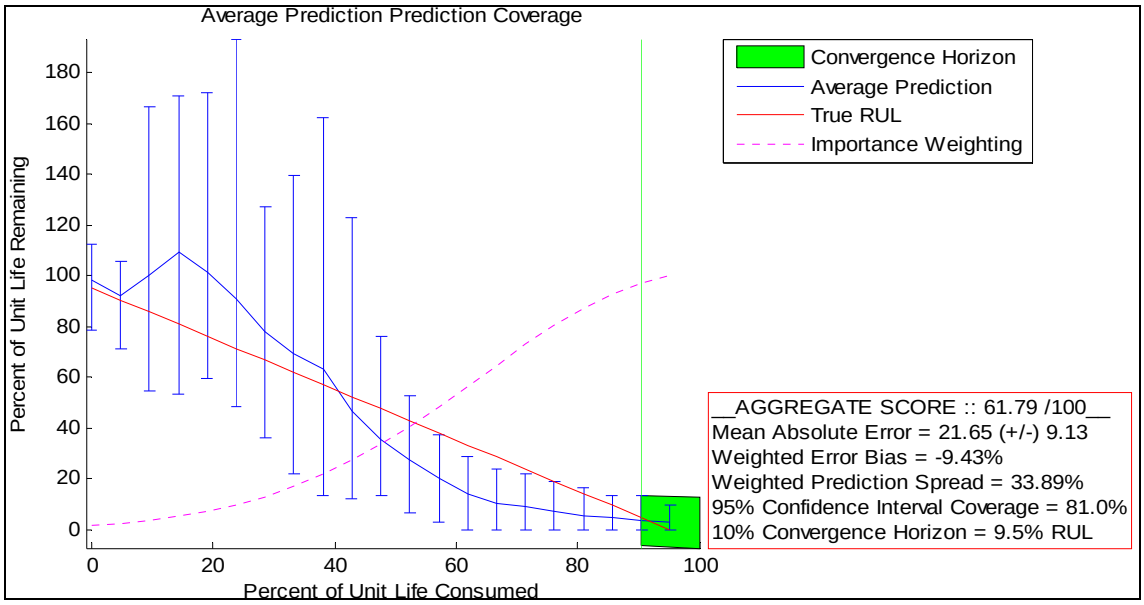


Figure 9.1-15: Best GA Model Using First 6 Features

The first thing to notice about the figure is that the deviation in the confidence intervals of the predicted RUL is quite large at 10-40% of life consumed, as in the previous figure. This large spread is due to how the GA generates the prognostic parameter, which is usually noisier than the OLS parameters. This model had a lower aggregate score and very high MAE than the OLS models. Again, further smoothing of the prognostic parameters before they are used in the GPM can result in better predictions for RUL. The OLS method produced better prognostic models than by using the GA method since the GA models had large MAE scores and lower aggregate scores.

Lastly the effect that occurs when more data points are used when extracting the features for prognostic parameter development was examined. As initially only 1 data point was used in the feature extraction process per motor test, 4 and 10 data points were also examined to see if there

was any great change in model results. Table 9-5 shows the results for the OSL method when 4 data points are used, with the original values shown for comparison. As the GA models generally had poorer results than the OLS models. Only the OLS models were considered when using more data points for feature extraction.

Table 9-5 Effect of Using More Data in Feature Extraction

1-Data Point in Features Extraction						
Features selected	**Aggregate score**	**MAE**	**WEB [%]**	**WPS [%]**	**Confidence interval coverage [%]**	**Convergence Horizon [%]RUL**
[1;16]	68.76	4.99+/-4.10	-1.02	23.95	100	0
[1;14]	68.26	6.02+/-4.14	-1.78	25.17	100	0
[1;6]	68.13	5.51+/-2.16	-2.16	20.56	95.2	0
[7;14]	66.79	11.37+/-2.76	-4.75	28.09	100	0
[1 4 5 6 9 12]	66.95	9.45+/-3.49	-3.39	28.79	100	0
4-Data Points in Features Extraction						
Features selected	**Aggregate score**	**MAE**	**WEB [%]**	**WPS [%]**	**Confidence interval coverage [%]**	**Convergence Horizon [%]RUL**
[1:16]	68.14	33.07+/-15.43	-2.70	24.72	100	0
[1:14]	70.64	36.48+/-17.12	-5.20	21.77	100	9.5
[1:6]	61.08	35.35+/-10.55	-5.65	16.68	66.7	0
[7:14]	62.16	69.87+/-10.11	-2.31	49.07	100	0

[1 4 5 6 9 12]	62.02	55.13+/-14.54	-4.72	32.93	85.7	0

The first thing to notice in the table is that the model built using features 1-14 has a higher aggregate score than the original results and is able to converge to the 10% RUL convergence coverage horizon. However, the MAE for this model and all other models developed by using four data points in feature selection have nearly a 30-58% increase when compared with the previous models. It can be concluded from this method that using more data points when extracting the features does not offer better prediction models in general. This effect is seen in the models built using features 1-16; the OLS model that used only one data points in feature extraction had nearly the same metric values but roughly 30% less MAE. Figure 9.1-16 shows the prognostic model outputs for the OLS model that used features 1-14 with four data points for each test.

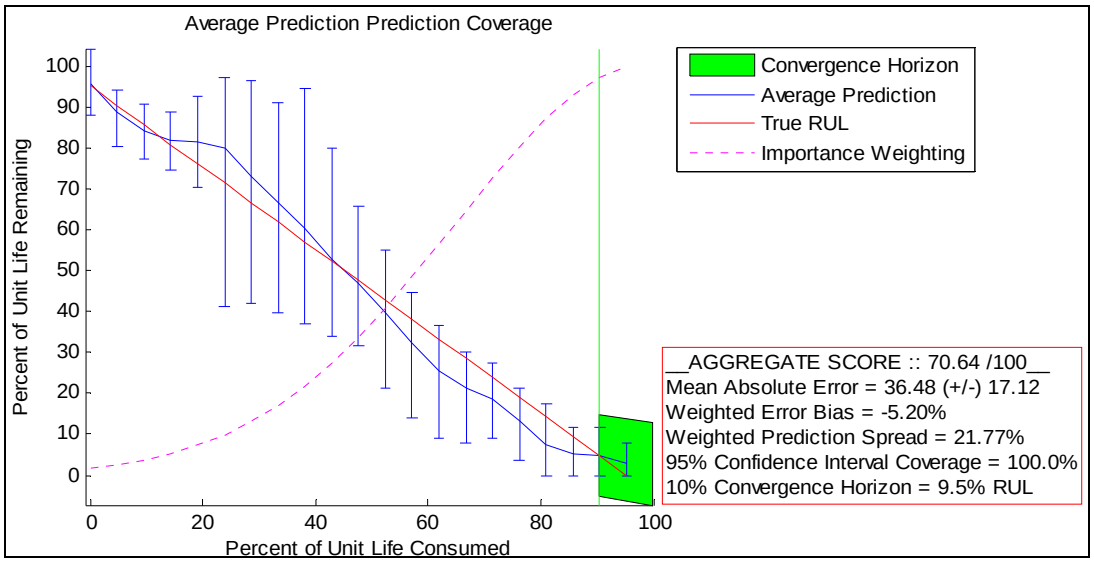


Figure 9.1-16: Performance Metric Plot Output Using 4 Data Points in Feature Extraction

It can be seen that the same problems of the prediction coverage and confidence intervals from 20-40% of life consumed occur. This model outperforms the previous OLS model in that the aggregate score is slightly higher and this model is able to converge at the 10% RUL convergence horizon, but the very high MAE value of the model is not desirable because of the larger error in the predictions. Table 9-6 shows a comparison of the model results in terms of performance metrics, obtained by applying the Ordinary Least Square Estimation to generate the prognostic parameters for the different combinations of features with ten and one data points for each test.

Table 9-6 Prognostic Model Outputs Using 10 Data Points in Feature Extraction

1-Data Point in Features Extraction						
Features selected	**Aggregate score**	**MAE**	**WEB [%]**	**WPS [%]**	**Confidence interval coverage [%]**	**Convergence Horizon [%]RUL**
[1;16]	68.76	4.99+/-4.10	-1.02	23.95	100	0
[1;14]	68.26	6.02+/-4.14	-1.78	25.17	100	0
[1;6]	68.13	5.51+/-2.16	-2.16	20.56	95.2	0
[7;14]	66.79	11.37+/-2.76	-4.75	28.09	100	0
[1 4 5 6 9 12]	66.95	9.45+/-3.49	-3.39	28.79	100	0
10-Data Points in Features Extraction						
Features selected	**Aggregate score**	**MAE**	**WEB [%]**	**WPS [%]**	**Confidence interval coverage [%]**	**Convergence Horizon [%]RUL**
[1:16]	67.09	117.16+/-50.84	-2.06	29.56	100	0
[1:14]	65.44	138.86+/-56.99	-4.82	28.64	95.2	0
[1:6]	59.81	132.87+/-39.36	-7.29	20.16	66.7	0
[7:14]	46.53	187.15+/-91.23	11.01	102.88	100	0
[1 4 5 6 9 12]	60.51	191.01+/-38.16	-6.59	32.32	81	0

In the table the model using features 1-16 has the best aggregate score, but it is lower than original OLS models. The MAE values for the new models are worse than the model results

obtained using four data points for each test in features extraction. Furthermore, the MAE values are nearly 100-150 higher in value than the original models. Finally, Figure 9.1-17 shows the graphical output of the best OLS model obtained using features 1-16 with ten data points for each test to generate prognostic parameters.

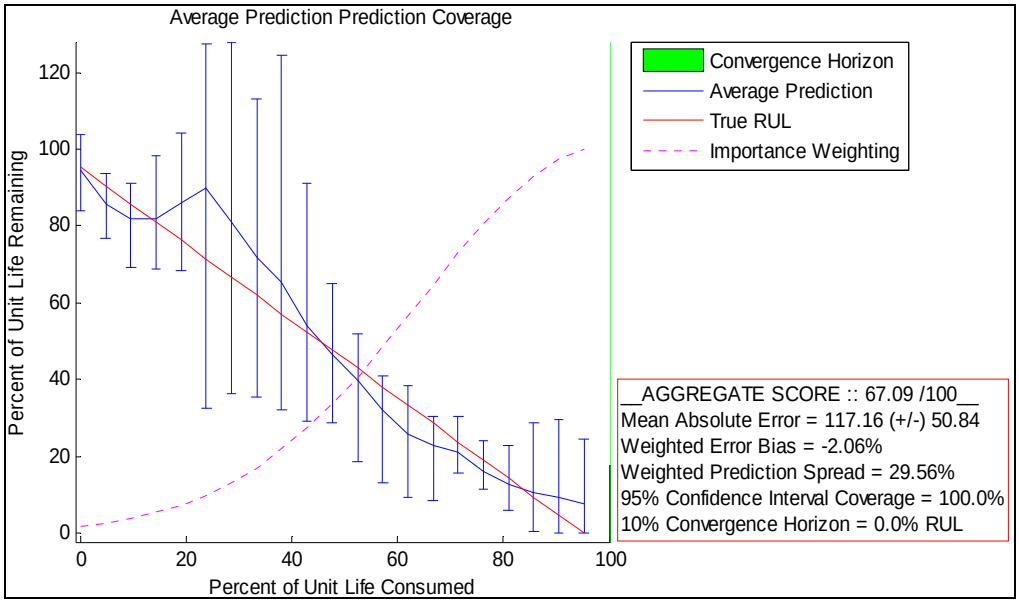


Figure 9.1-17: OLS Model Results Using 10 Data Points for Feature Extraction, Features 1-16

Looking at the figure, it can be seen that the same problem of large deviations in the RUL predictions also occurs in this model. The other thing of note is the large 118% MAE metric. This error is too large to be considered useful, so only 1-4 data points were used in all models.

9.2 Large Impeller Degradation Experiment

In total there were 28 impellers tested during this experiment. Four of the tests were excluded due to external factors corrupting the test procedure, such as a power failure that rendered the pump unable to re-prime after some amount of testing. The work completed involved developing the three different prognostic model types using the pump failure data, which will be the focus of this current report. In the first subsection the results for the Type I model are shown, followed by the results for the Type II and Type III models.

In each of the following subsections, the data and processing needed for each of the model types is shown. The end result of each of these models is to obtain RUL estimates for the system at a current time of operation or at a certain level of component degradation. Furthermore, the transition from the three different model types should also increase the accuracy of the RUL

estimates and reduce the associated uncertainty. All residual and model development was performed using the PEM and PEP toolboxes.

9.2.1 Type I Prognostic Model Results

There were a total of 28 impellers that were tested during the course of the experiment; four of these tests were omitted due to premature failures. Additionally, during model development 6 more tests were excluded due to unusable features, this will be discussed more fully in the Type III model development section. There was a wide range of failures times for the remaining 18 tests, from 1 to 33 days. In a Type I model, the only information that is used is the failure times for each of the tests, these are fitted to a distribution and RUL estimates for the current time of the system can be calculated. The first step in the analysis is to fit the failure time data to a distribution. This helps determine which model fit will work best for the supplied data. In Figure 9.2-1 an exponential fit to the failure time data is shown.

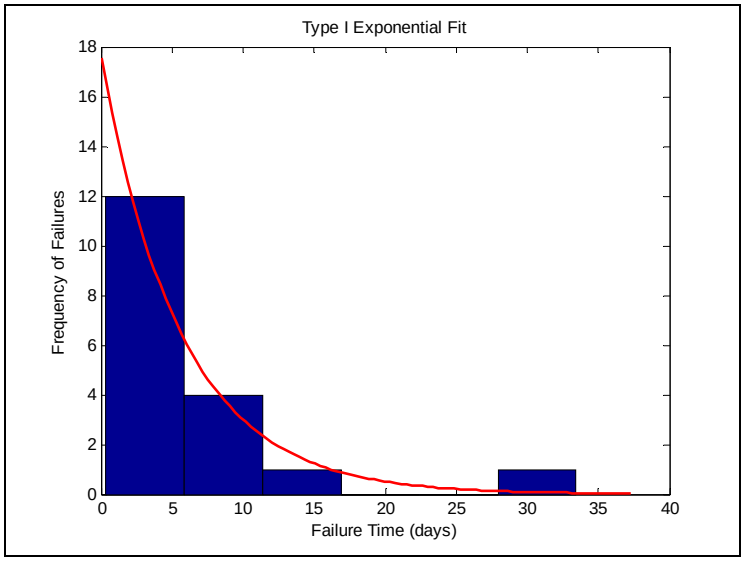


Figure 9.2-1: Exponential Fit to Failure Time Data

In the preceding figure, it can be seen that the majority of the failures occur between 1 to 15 days. This portion can be termed to have a fast failure mode. Beyond 15 days, the remaining units fail around 4-5 weeks, which is a second, slower failure mode. The next step is to develop several Type I models by using a leave one out cross validation method. This means that when the RUL estimates are made, one of the failure times is left out during model development. This process is looped until all failure times have been left out, this process results in 18 separate RUL estimates computed for various percentages of system life. These results are averaged along with the ToF and the Median Absolute Percent Error (MdAPE) is calculated for each percentage unit of life.

All Type I models developed used Weibull distributions. In Table 9-7 the MdAPE for the model ToF estimates is shown.

Table 9-7: Type I Model ToF Estimates

Model/% Life	10%	20%	30%	40%	50%	60%	70%	80%	90%
Type I Weibull	62.3	63.6	56.3	54.4	64.1	76.2	88.2	100.1	111.8

In the table, the error begins to decrease, but after 40% of life the error begins to rise. Ideally, the error should be large at the start of life and decrease at the end of life. The reason the error increases is due to some outliers in the data from 24-33 days. All other failure times were from ~1-15 days. Next, the development and results for the Type II prognostics model are shown.

9.2.2 Type II Prognostic Model Results

If the operating conditions for the system under consideration are known, then these conditions can be incorporated into a Type II prognostic model. For this research a Proportional Hazards Model (PHM) was chosen because the operating conditions simulated by closing the ball valve in the outlet line of each pump by different increments should be additive or multiplicative when compared to some baseline condition.

For this type of model additional data is needed that is not required for a Type I model. This data also must be placed into a specific structure so that an accurate model can be developed. The first piece of information needed is the operating conditions of the experiment; in this case it is the different ball valve positions. These are termed the covariates of the model and reflect how the different operating conditions stress the particular component in question. Next, the failure times are required, but these need to be separated into their respective operating condition. For example, all the tests that used the 100% open condition have their respective failure times and so on. The next data needed is the frequency of each of the failure times for each operating condition. In this data, the 100% condition had four failure times that lasted five days, so the frequency associated with the five day failure time is four. Finally, any test that needs to be censored or not is given a value of 1 or 0, respectively.

Once all this data is collected, the PHM can be developed. The first step in development is to check if the reliability functions for each operating condition are proportional in some way. This

proportionality check is accomplished by taking the log of the negative log of the reliability functions. In Figure 9.2-1 the reliability function for each operating condition is shown in the left figure and the proportionality check is shown in the right figure.

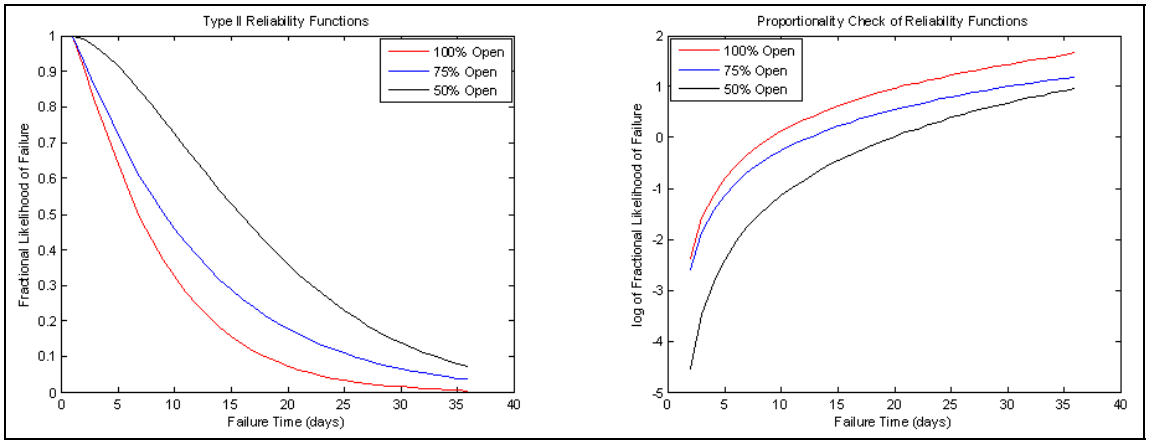


Figure 9.2-1: Type II Reliability Functions & Proportionality Check

The reliability functions for the 100% and 50% open operating conditions show ideal proportionality while the 75% open operating condition is skewed at the start and later in life. The reason that the 75% condition is not as proportional to the other two can be attributed to the fact that this condition had several failures at 1 day and one failure at 33 days, while the majority of failures in the other two operating conditions were on average the same. Next, the same system times are used as in the Type I models to obtain RUL estimates and average uncertainty. In Table 9-8 the MdAPE estimates for the PHM are shown along with the Type I models.

Table 9-8: Type I & II Model ToF Estimates

Model/% Life	10%	20%	30%	40%	50%	60%	70%	80%	90%
Type II PHM	84.83	71.88	62.82	52.38	41.67	29.45	25.54	16.29	7.84
Type I Weibull	62.3	63.6	56.3	54.4	64.1	76.2	88.2	100.1	111.8

In the table, the PHM error results perform as expected; they are large at the start of life and decrease to below 10% near the end of life. These results also demonstrate how the inclusion of additional information about the system can lead to better RUL predictions with a small amount of error.

9.2.3 Type III Prognostic Model Results

In this section the data processing needed for feature extraction, prognostic parameter generation and GPM development and results will be shown. This last type of prognostic model considers the stress or degradation that the component experiences over lifetime, and if this degradation can be tracked or measured then this information can be incorporated in the development of the GPM. The RUL estimates of this model use test degradation paths and extrapolate using a functional fit to some predetermined failure threshold. How long it takes to go from the end of the path to the threshold is the RUL for that component. The most difficult part of the Type III model development is the identification and combination of degradation paths that appear in most or all failure data. With this in mind, the first part of this section will focus on feature extraction methods examined in the time and frequency domain, as well as feature extraction from an AAKR model. Next, the prognostic parameter generation and suitability metrics are briefly discussed followed by the use of the failure and test prognostic parameters in the development of the GPM and RUL estimate results.

9.2.3.1 Signal Feature Extraction Methods

The first method of feature extraction is to calculate statistical measures of the raw signals. The measures included the mean, standard deviation, variance, skewness, kurtosis and Root Mean Square (RMS). These calculations were performed in a windowed fashion, using the sampling frequency at 1024 Hz as the window length; this reduces the large amount of data into a single value for each recording period. The differential pressure signals for almost all tests showed good trends for several of the measures, while the vibration and current signal features were found to be unusable for all measures due to no useful trends. An example of the RMS values for the differential pressure signals for several tests is shown in Figure 9.2-2.

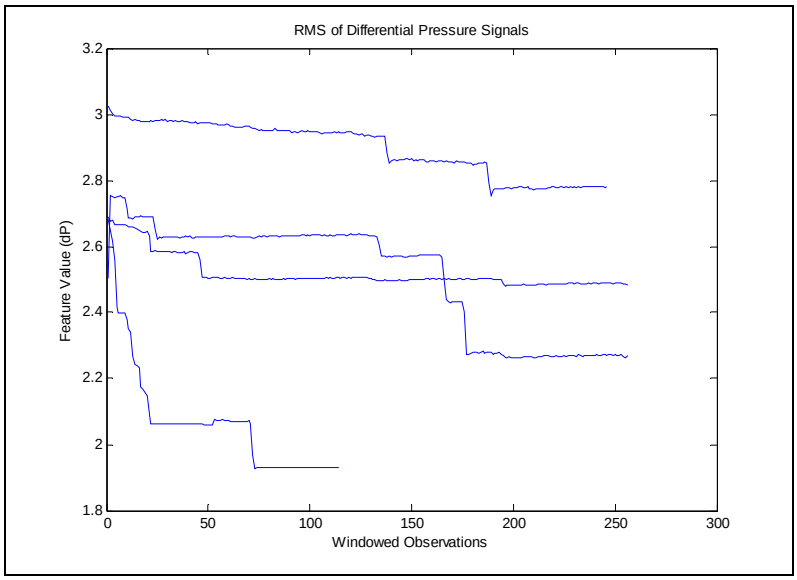


Figure 9.2-2: RMS Feature Extracted from Differential Pressure

In the figure, the step changes in each plot indicate the point when one or more impeller blades had snapped off. This degradation decreases the pressure in the pump housing and is captured by the signal and feature. These features could be combined with others that show similar trends. However neither the vibration nor current features exhibited the trends shown in the figure. The next step is to examine these signals in the frequency to determine if any of the extracted features can be combined to form a useful degradation measure.

In the frequency domain, the Fast Fourier Transform (FFT) was utilized as a feature extraction method. Each of the signals is windowed using the same window length of 1024 samples, and a FFT is calculated for each of these windows. The peak values for certain frequencies are then examined for the FFT of an unfailed window and compared against the FFT of a window before failure. The changes in the maximum peak values for each of the frequencies of interest are then tracked through the entire lifetime of the test. This method was performed on all tests and signals, but none of the frequency features showed any usable trend that could be combined with the time domain features.

The last method to extract features from the raw data was to use an AAKR model and generate failure residuals. In this method, the first 1024 samples of each test are used as training data for that particular pump since none of the tests under consideration showed any degradation or failures at the start of testing. The rest of the dataset is then considered as failure data. This failure data will be used in the AAKR model and the residuals examined. An example of model failure residuals from one pump test is shown in Figure 9.2-3.

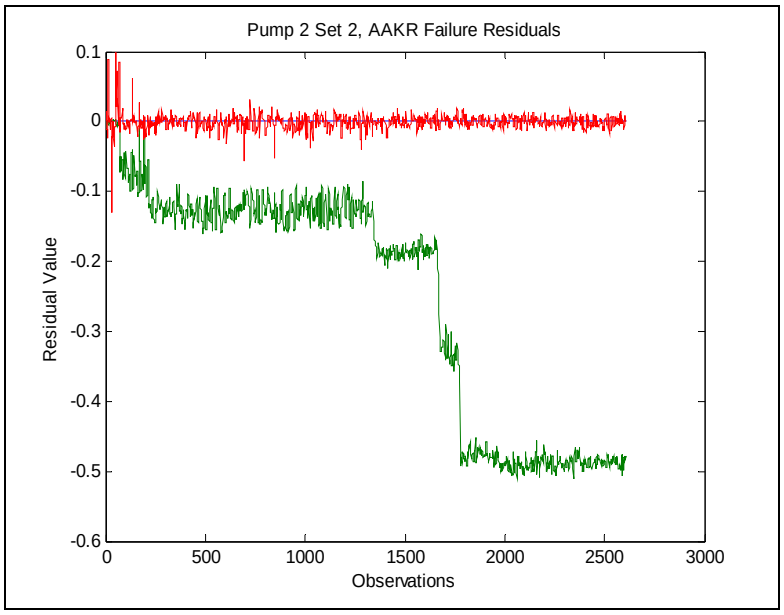


Figure 9.2-3: AAKR Model Failure Residuals Pump 2 Set 2

In the figure, the green plot is the differential pressure signal residual and the red plot is the current signal failure residual. The vibration signal is also included but the residual value is very small and is covered by the current signal, this residual has the same trend as the current signal residual. The differential pressure residual has the same shape as seen in the time domain features and can be combined to form a prognostic parameter. Other methods of feature extraction were explored, including the Joint Time Frequency Spectrum and the Hilbert-Huang Transform; however these methods did not yield any usable features as those seen in the time domain or AAKR modeling. The next step is to combine the features into a prognostic parameter so that the GPM can be developed.

9.2.3.2 General Path Model Development

The prognostics parameters were developed using the time domain features. Development of the parameters was carried out by using a genetic algorithm optimization function; this is contained in the PEP toolbox and by use of OLS regression. Of the 24 total tests, only 18 were found suitable for use in modeling because many of the failure residuals were small in magnitude or did not show as noticeable trends as seen in the previous figure. As stated in previous reports, there are suitability metrics that are used for the generated prognostic parameters, which are monotonicity, prognosability and trendability. Monotonicity measures if the parameter is generally increasing or decreasing in value of time monotonically; prognosability measures how

useful the parameter will be for modeling; trendability measures how well the parameter can be fit to a functional form. All of these suitability metrics are measured from 0 to 1, with a value close to 1 being desirable. In Table 9-9 the metric values for the parameters generated using the GA and OLS are shown.

Table 9-9: Performance Metric Values for Prognostic Parameters

Metric	G.A Parameters	OLS Parameters
Monotonicity	.4547	.2654
Prognosability	.4738	.8674
Trendability	.2644	.0084

In the table, the trendability metric for the OLS case is very low, this low value is due to the noise of the parameters, which is difficult to trend. The shape of the parameter also accounts for the average monotonicity values for both cases. Finally, the prognosability value for the OLS case is very high, due to the linear shape of the parameters. In contrast, the GA parameters have a step shape. In Figure 9.2-4 the GA parameters are shown.

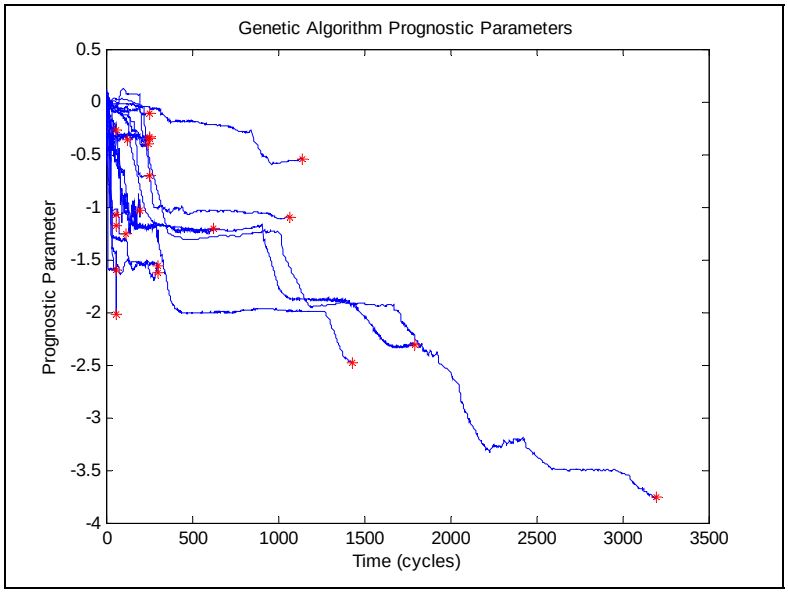


Figure 9.2-4: Failure & Test Case Prognostic Parameters

In the figure, the GA parameters have a wide range of ending failure times as well as a wide range of degradation values. Since the resulting parameters are not usable in a GPM because of these wide ranges, the OLS parameters were developed and investigated. In Figure 9.2-5 the OLS parameters are shown.

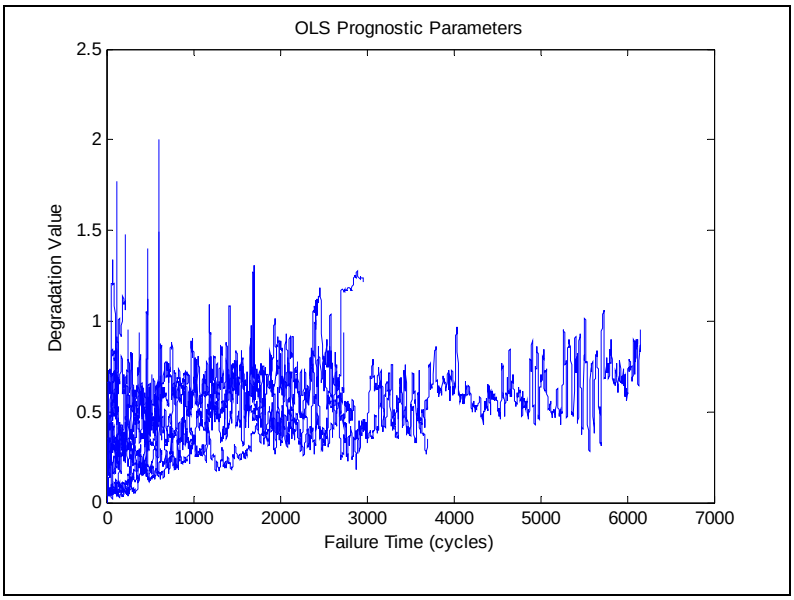


Figure 9.2-5: OLS Prognostic Parameters

The OLS parameters shown have in general a linear shape and were used for further model development. Since the OLS parameters have the same problems of large ending times and degradation values as the GA parameters, it was decided to set a soft failure threshold at .75 degradation units or 75% failed. Any data after this threshold is removed because the pump is considered as operating in a failed condition. The failure times are then redefined as the end of these new cut parameters. The Type I and Type II models were developed and validated using these modified failure times. In Figure 9.2-6 the new modified OLS prognostic parameters are shown.

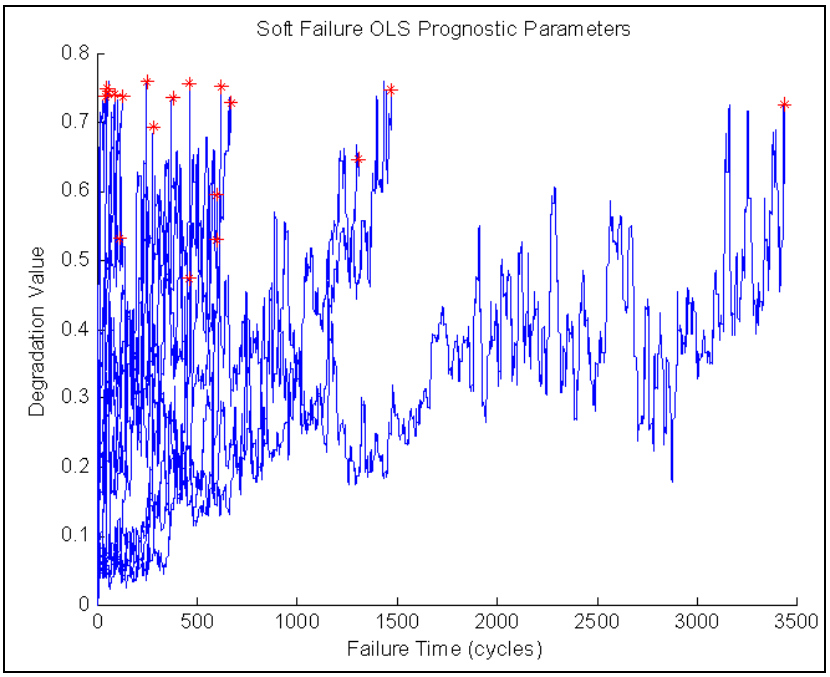


Figure 9.2-6: Soft Failure OLS Prognostic Parameters

The new parameters shown in the figure still have a wide range of ending times, but now the majority of the degradation values fall between .5-.75, which will help reduce error during model development. Table 9-10 gives the MdAPE values for all models developed.

Table 9-10: All Prognostics Model ToF Error

Model/% Life	10%	20%	30%	40%	50%	60%	70%	80%	90%
Type III GPM	57.31	51.07	45.22	38.02	31.50	24.29	18.17	10.94	5.25
Type II PHM	84.83	71.88	62.82	52.38	41.67	29.45	25.54	16.29	7.84
Type I Weibull	62.3	63.6	56.3	54.4	64.1	76.2	88.2	100.1	111.8

In the table, the error seen for the GPM is lower than the PHM. This model also behaves as expected by having large errors at the start of life and a low error near the end of life. Again, the

reduction in error demonstrates how the inclusion of degradation information can further refine the RUL estimates and reduce error. In Figure 9.2-7 the MdAPE values over lifetime are shown for all developed models.

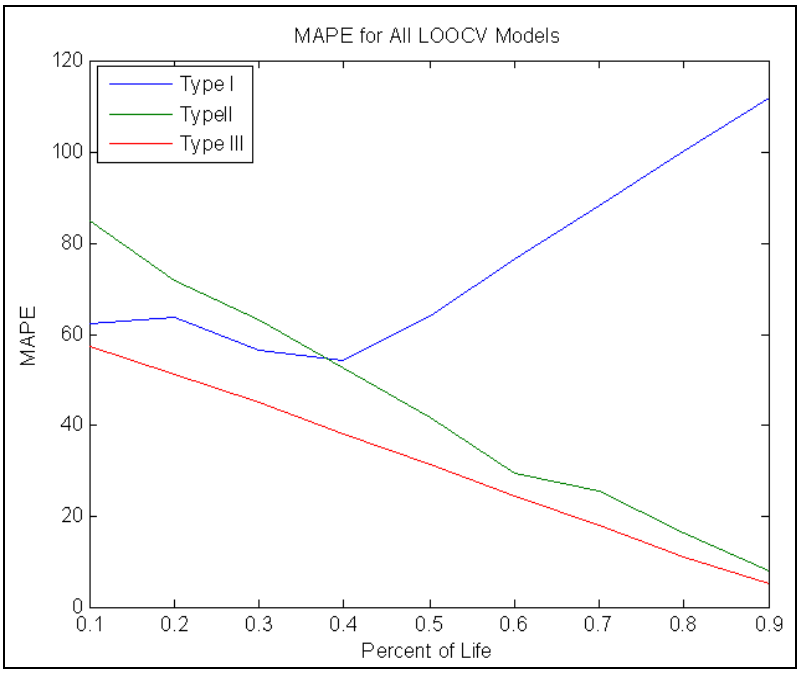


Figure 9.2-7: ToF Errors for Developed Models

In the figure, the Type I model shows the highest error, while the Type II and III models both show a decreasing error in the ToF estimates over the lifetime. This decrease in error is the desired result. With more failure cases these models can be refined so that the error is less than shown here.

9.3 Analysis of Heat Exchanger Data

To determine an optimal lifecycle prognostic method, multiple competing models were created. Four signal sets were selected to build auto-associative kernel regression models and ordinary least squares regression of each residual set was used to produce prognostic parameters. For the General Path Model (GPM), a linear and quadratic fit was used for each case, and Bayesian updating was applied. All models developed utilize degradation data collected at a 1 gallon-per-minute operating condition.

9.3.1 Signal and Feature Sets

The first set of calculated features include the log mean temperature difference (LMTD), heat rate, and delta temperatures. The two features used in the prognostics models are heat rate and

overall heat transfer coefficient given by Equation 9-8 and Equation 9-10 respectively.

$$\dot{Q}_{\frac{h}{c}} = \dot{m} C_p (T_1 - T_2)$$ **Equation 9-8**

$$\text{LMTD} = \frac{(T_{h1} - T_{c2}) - (T_{h2} - T_{c1})}{\log\left(\frac{T_{h1} - T_{c2}}{T_{h2} - T_{c1}}\right)}$$ **Equation 9-9**

$$U_{\frac{h}{c}} = \frac{\dot{Q}_{\frac{h}{c}}}{LMTD * A}$$ **Equation 9-10**

where A is the surface area of heat transfer.

These signals and features define the state of the system and are selected for inclusion in the development of an AAKR model. As mentioned previously in this document, the AAKR model requires training, testing and validation data. When cleaning the training data for the AAKR model, it is important that the data is fault-free and the test cases operate in the same conditions. To reduce system noise, especially for the mass flow rates, a median filter was applied to remove outliers exceeding three standard deviations. This procedure removed many of the large spikes seen in the mass flow rate signals.

It is also important to develop AAKR models with groups of related variables. Therefore, the linear relationships between the signals and features were analyzed via correlation coefficients. Absolute coefficient values of greater than 0.7 correspond to strong correlations between signals, and coefficients of 0.25 and below are considered to show no significant linear correlation. Figure 9.3-1 shows a plot of the correlation coefficients of the raw data and calculated feature indices, with indices summarized in Table 14-1.

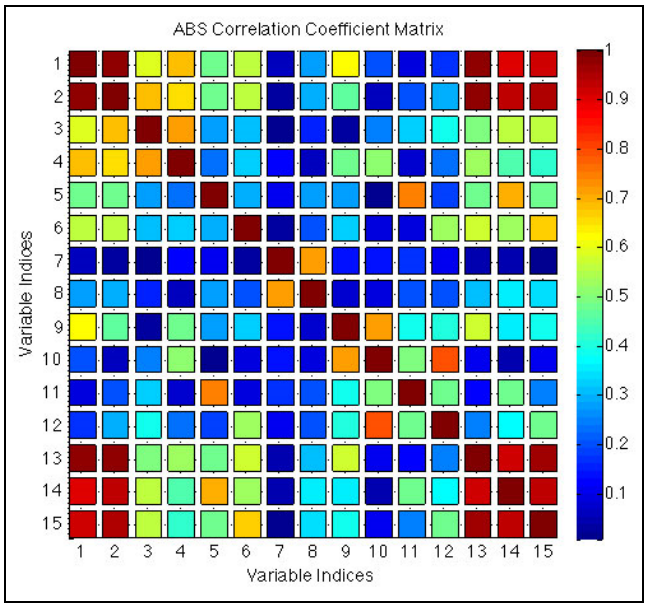


Figure 9.3-1 Correlation coefficients of signals and features

The figure above shows that there is a strong correlation between signal indices 1 to 4, which are the measured temperatures. There is also a strong correlation between signals 1 and 2 and features 13 to 15, these are respectively the LMDT and heat transfer coefficients. There are moderate correlations between signals 1 to 6; 5 and 6 are the flow rates; and 13 to 15.

Four sets of related variables were chosen based on correlation coefficients and understanding of the system processes. Other signal sets were tested during initial modeling, but did not return desirable residual values and trends, therefore they were not considered for final lifecycle prognostic models. The selected signals and features were chosen either for being moderately-to-highly correlated to one another or for the strong trend observed. The indices chosen for each signal set are given in Table 9-11.

Table 9-11 Signal sets used for modeling

Signal Set	Signal/Feature Indices Used
1	2, 3, 11, 12, 14, 15
2	1, 2, 3, 4, 11, 12, 14, 15
3	1, 2, 3, 4
4	1, 2, 3, 4, 14, 15

In signal sets one, two, and four, the heat transfer coefficients, heat rates, and temperature signals are used. Since the overall heat transfer coefficients (indices 14-15) are calculated from first principles models that are dependent on temperature signals, including them in an empirical AAKR model has the effect of increasing both the models' and prognostic parameters' weightings toward the temperature signals. This may improve modeling attempts when the temperature signals have strong increasing trends, and is expected to be more effective than other methods of artificially increasing the weightings.

9.3.2 Auto-Associative Kernel Regression Model

During pre-processing, the unfaulted heat exchanger data is divided into three data sets termed training, testing, and validation. Training data is used to train the model and should consist of unfaulted data that covers the range of operating values. Testing data is used for bandwidth optimization and performance metric generation, and validation data is used to validate the performance ability of the model. AAKR models for the heat exchanger were developed and evaluated with the PEM toolbox. Kernel regression requires a parametric kernel function, in this case a Gaussian function, defined by a bandwidth that specifies the region of localized weighting for an input vector to the memory matrix output. An optimal bandwidth can be selected by altering it to minimize the error between known unfaulted observations and the model output. This method of determining the bandwidth increases the accuracy of the kernel regression model. The training residuals from an AAKR model of signal set 2 are shown in Figure 9.3-2.

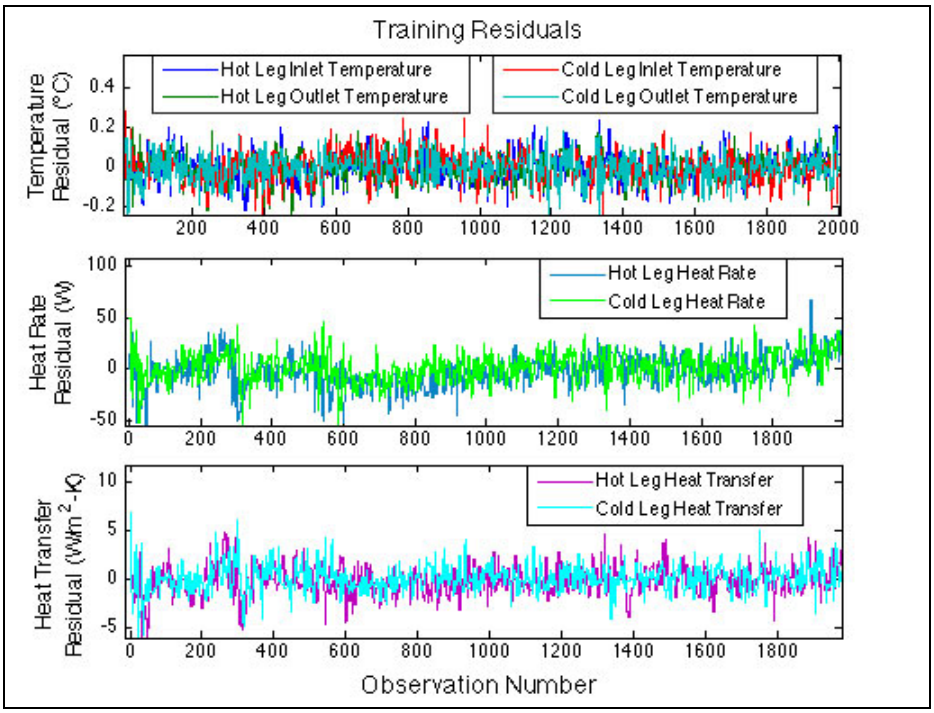


Figure 9.3-2 Training residuals for signal set 2

For this experiment, the training residuals of the temperature signals are desired to be less than 1° C since the temperature signals change less than 10°C over the faulted range. The training residuals of the heat rate should optimally be less than 50 W, and the heat transfer coefficient residuals should be less than 10 W/m^2 K. These levels were chosen based on knowledge of signal and feature operating ranges over normal cycles. After the model is built, faulted data is passed through and residuals for each faulted cycle are calculated. An example of faulted residuals for the temperature sensors of signal set 2 is plotted in Figure 9.3-3.

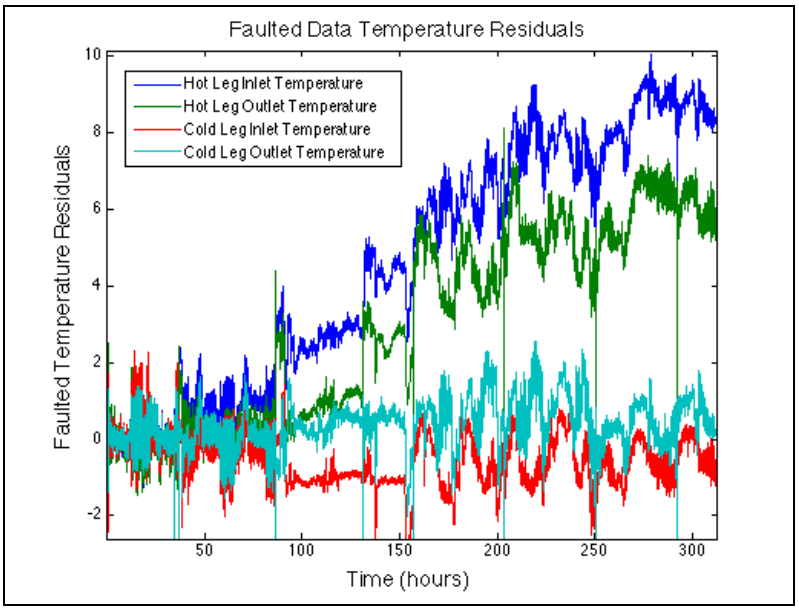


Figure 9.3-3 Faulted residuals of temperature signals (indices 1-4) using signal set 2 model

From the failure residuals shown, strong increasing trends can be seen for the hot leg temperature signals. Dominantly monotonic trends are important when combining residuals to make a prognostic parameter. When combining the residuals, the objective is for the resulting health indicator to increase or decrease over the lifecycle to help indicate the degree of system or component degradation. If the observed trends of the residuals show a strong increasing/decreasing trend then the resulting prognostic parameter will also have a strong trend and be more useful for RUL predictions.

9.3.3 Prognostic Parameter Generation

The prognostic parameter is a single metric of the amount of deviation from normal behavior of the system and is ideally linked to the overall health of the system. In this project, it is calculated as a linear combination of the residuals from the AAKR model. While a genetic algorithm may be used to find a linear combination of weights for the residuals, the algorithm is computationally expensive. Instead, an OLS regression is applied that mimics the optimization and is less computationally intensive for smaller data sets. The monitoring model residuals of multiple runs to failure are collected into a single matrix by concatenating each test case. This creates an **n** x **s** matrix, **X**, where **n** is total data points in all test cases, and **s** is the number of signal residuals output from the model. This **X** matrix is regressed against the **n** x **1** vector **y** where each $\mathbf{y}_i$ corresponds to the percent of the total unit life at that observation. This means that the residuals of each test case are fitted to a linear curve from 0 to 1. The linear weights are then

$$\hat{\beta} = (X^T X)^{-1} X^T y \qquad \textbf{Equation 9-11}$$

where $\hat{\beta}$ is an **s** x 1 vector.

9.3.4 General Path Model and Bayesian Updating

When using the GPM approach, a parametric function is fit to the degradation parameter and extrapolated until it crosses a predefined failure threshold. Typically, the failure threshold is based on historical failures but need not directly indicate a point of catastrophic failure. The failure threshold can be set as any point where a system no longer conforms to the necessary specifications and demands placed upon it.

Because of the limited number of test cases, the GPM and all components are created by the use of a leave one out cross validation (LOOCV) technique. Hence, to calculate the RUL of a specific case, every other case is used to build the model. This avoids invalidating a model by keeping training and testing data separate yet general enough to compare over all cases. With more data, an alternative approach could be to simply divide the cases in half and build one model. The degradation path is assumed to have the general linear form:

$$Y|\beta, X, \sigma^12 \sim N(X\beta, \sigma^12\ I) \qquad \textbf{Equation 9-12}$$

where **y** is the response a vector, **X** is the input data matrix, and **β** is the vector of regression parameters. This model assumes normally distributed errors with variance $\boldsymbol{\sigma^2}$.

Development of failure thresholds had to be generated with respect to the data. The values were chosen as a reflection of an unacceptable amount of degradation, limited by the least degraded cycle for any given model. Any data collected after this point was considered past failure and removed from the data analysis. A histogram plot of the failure times used in the development of the lifecycle prognostic models is shown in Figure 9.3-4.

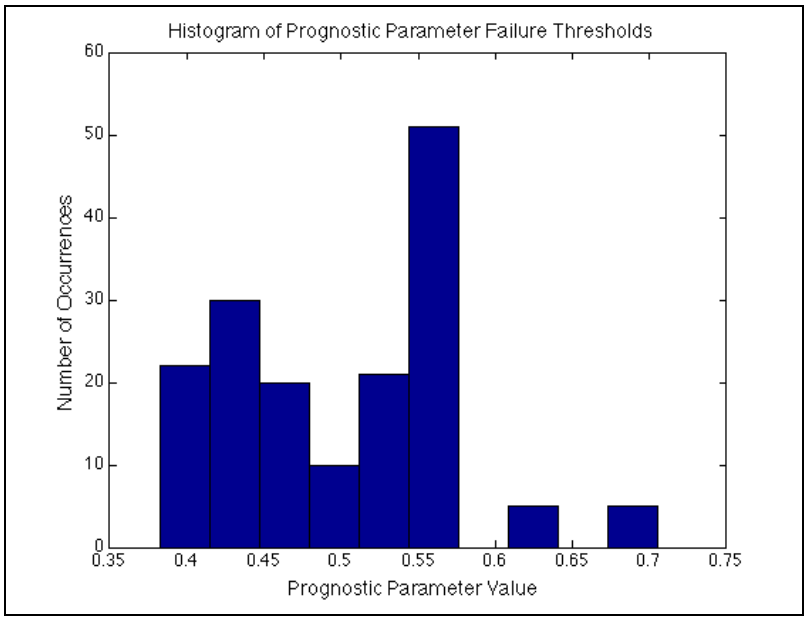


Figure 9.3-4 Histogram of failure thresholds

If the test case data is censored such that only data before a time step is available, then the RUL can be calculated at each time step by extrapolating the current degradation path to the failure threshold. To do this, a suitable parametric fit must be chosen. The fit can be of any linearly separable functional form such as linear, quadratic or exponential. The OLS method is used for regression of the parametric fittings because the OLS regression on a joint Gaussian distribution of parameters gives the maximum likelihood estimate. This method assumes that the error is normally distributed around zero. The OLS solution can be found using the pseudo-inverse given previously in Equation 9-11.

By adjusting the functions in the columns of the input matrix **X**, different fits can be applied to any test path. It is assumed that for a certain failure mode the degradation paths will follow similar fits. Therefore once a suitable fit is chosen for the failed data, it is assumed the censored faulted data will follow the same fit.

Bayesian priors can also be incorporated into the OLS model to reduce the uncertainty and increase the stability of RUL estimates. Bayesian statistics combines prior distributions with sampled data to create a posterior distribution. If few data points are available during model development, then without incorporating any form of Bayesian prior estimations the model can easily be affected by noise and give widely varying predictions of time to failure. Bayesian methods can be used to incorporate prior knowledge of regression parameters in the GPM. This approach requires historical run-to-failure data in order to evaluate the prior distributions of regression parameters. An alternative approach instead uses RUL estimates from Type I

prognostic models as prior information. In this approach, the Type I RUL distribution is treated as an additional data point in the OLS regression. The measured data are augmented with the distribution according to:

$$Y = \begin{bmatrix} y \\ thresh \end{bmatrix}, X = \begin{bmatrix} X \\ MTTF \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_y & 0 \\ 0 & \Sigma_{RUL} \end{bmatrix} \qquad \textbf{Equation 9-13}$$

where y is the observed prognostic parameter, *thresh* is the failure threshold, x is the timestamps , *MTTF* is mean failure time from the Type I distribution Σ_y is the noise or uncertainty associated with the observed prognostic parameter, and Σ_{RUL} is the uncertainty in the Type I RUL estimate. The OLS regression is then solved:

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y \qquad \textbf{Equation 9-14}$$

$$V\sigma^2 = (X^T \Sigma^{-1} X)^{-1} \qquad \textbf{Equation 9-15}$$

$$\sigma^2 = \frac{1}{n-k} \left(y - X\hat{\beta}\right)^T \Sigma^{-1} \left(y - X\hat{\beta}\right) \qquad \textbf{Equation 9-16}$$

where k is the degree of the parametric function used in the GPM.

The weight of the prior information in the OLS regression depends on two main factors: the variance of the prior relative to the variance of the data, and the number of observations collected. If the variance of the prior is small compared to the noise of the data, the prior β_0 will be weighed more heavily. However, no matter the difference in variance, with enough observations the data should eventually swamp out the prior in calculating the posterior.

9.3.5 Bayes Method Implementation

For each of the four AAKR models, two prognostic modeling methods are used:

GPM Method 1: No Bayesian updating

GPM Method 2: Type 1 Bayes priors

To compare the two methods, plots of the predicted TTF versus the actual TTF are examined. In each plot, the multiple blue lines correspond to the determined TTF of each cycle over time. Figure 9.3-5 shows the TTF comparison when no Bayesian updating is used when developing the GPM.

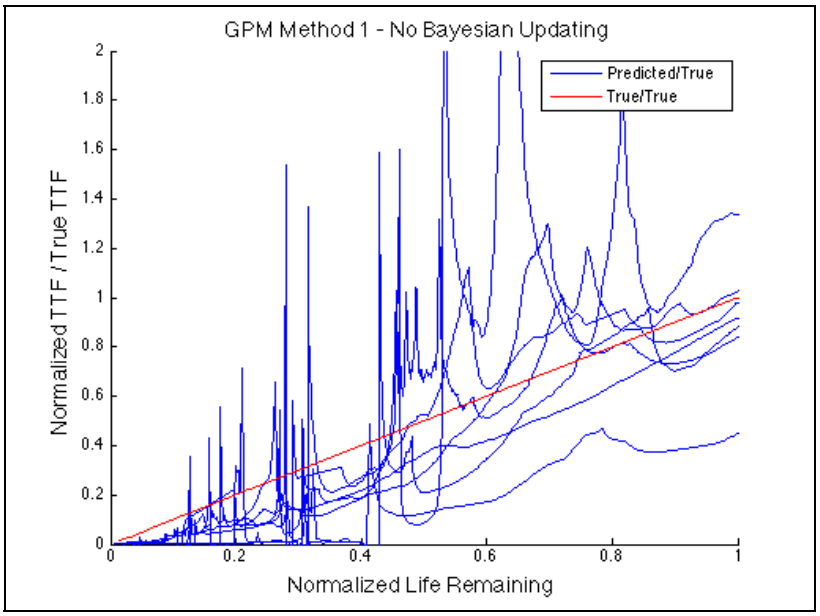


Figure 9.3-5 GPM method 1 TTF predictions across cycles without Bayesian updating

Without Bayesian updating, TTF prediction times have large spikes, and prediction accuracy is reduced. While some peaks are due to the noise and artifacts in the heat exchanger data acquisition system, the somewhat larger and broader peaks at regular intervals are most likely the result of the regular additions of clay into the hot fluid. The extra clay would change the thermodynamic properties as well as mass flows of the otherwise closed system. In an attempt to improve TTF estimation, past cycle failure times are incorporated as prior information (Type I) as shown in Figure 9.3-6.

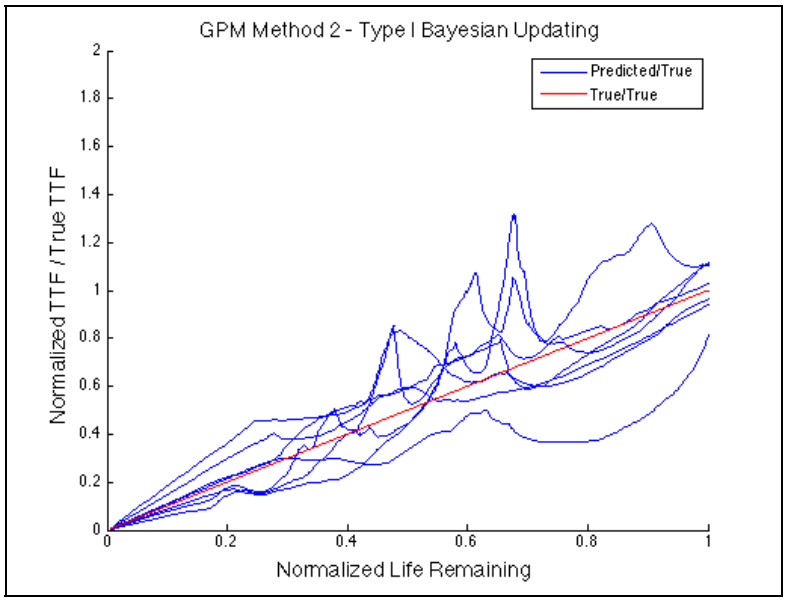


Figure 9.3-6 GPM method 2 TTF predictions across cycles with Type I Bayesian updating

The predictions using Type I prior information show visual improvement over those with no Bayesian updating.

9.3.6 Results and Discussion

Initial modeling attempts revealed that using a quadratic fit is more accurate than using a linear fit; therefore, to conserve space, results will be confined to quadratic fit models. The different GPM methods and signal sets (models) are compared using several performance metrics.The first model comparison metric used is the Absolute Error Mean (AEM), which returns the average absolute difference between the predicted RUL and the true RUL in real time units, shown in Figure 9.3-7.

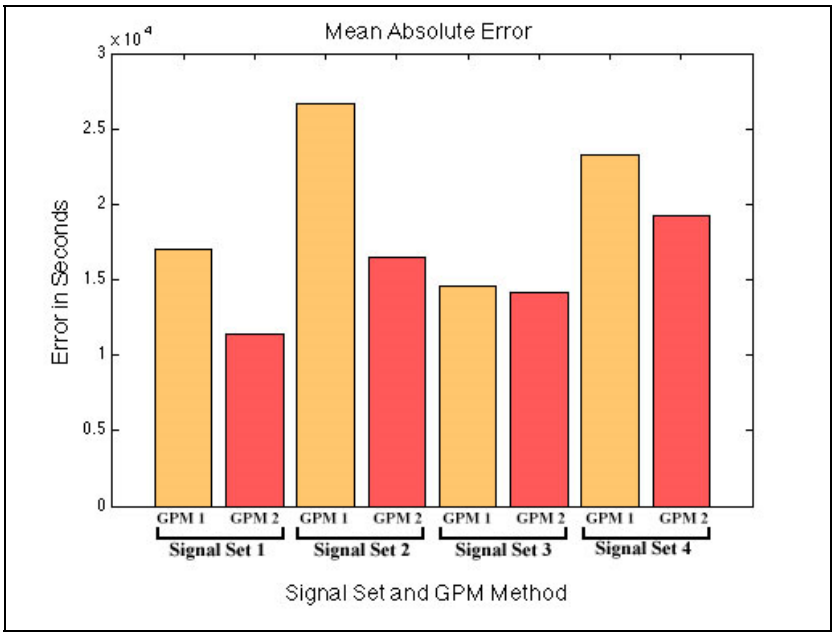


Figure 9.3-7 AEM for four signal set models and two GPM methods

Signal sets 1 and 3 have the lowest AEM, and GPM method 2 further improves the predictions. Signal set 1 with GPM method 2 results in the most accurate RUL predictions for this data set. The second metric used to evaluate the prognostic models is the Absolute Error Standard deviation (AES), which is a measure of the variation in error through time of each model and GPM method, shown in Figure 9.3-8.

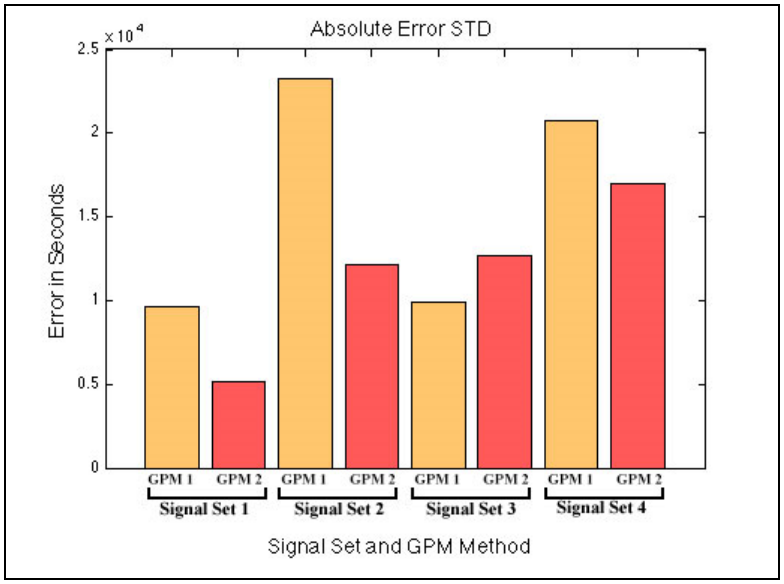


Figure 9.3-8 AES deviation for four signal set models and two GPM methods

Again, the model using signal set 1 and GPM method 2 shows the best performance, with highest precision in estimating the RUL. To quantitatively compare the different GPM methods, the AEM, AES, spread, and coverage metrics are used. A plot showing the results of these metrics for each GPM method for signal set 1 is shown in Figure 9.3-9 and the unnormalized metric scores are shown in Table 9-12.

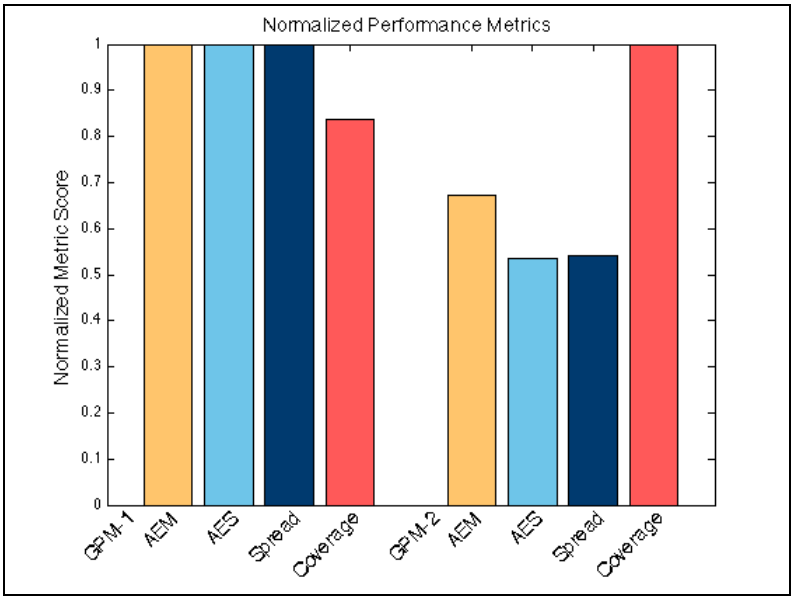


Figure 9.3-9 Plot of normalized performance metrics for two GPM methods and signal set 1

Table 9-12 Performance Metrics Scores

GPM-1	AEM	1.7026E4
	AES	9.6206E3
	Spread	131.135
	Coverage	83
GPM-2	AEM	1.1441E4
	AES	5.1395E3
	Spread	70.767
	Coverage	99

These metrics indicate that the Bayesian updating method (GPM Method 2) is more accurate for predicting RUL for this data set.

9.3.7 Finishing Efforts

The prognostic methods presented here can be improved in several ways. The noise of the prognostics parameter can be reduced by improved filtering or prognostics parameter optimization. An optimized prognostics parameter with a well-defined degradation threshold could increase the prognosability and decrease the end of life RUL and TTF prediction errors. Next, the application of a fault detection method to cut beginning of life test data before a fault is detectable could be implemented. Cutting data that is similar to clean or unfaulted data would increase trendability, particularly for linear GPM fits that would not accommodate a sudden shift in degradation. A mitigating factor to this is that all test cases are initially run with clay in the system, so physically some form of degradation is seen at the start of life.

9.4 Barkhausen Noise Measurements

As part of a recent collaboration between the Pacific Northwest National Laboratory and the University of Tennessee Nuclear PROaCT Research group, an investigation was performed evaluating the feasibility of prognostics on passive systems and equipment. Specifically, a data set monitoring the Barkhausen noise for a series of mechanically stressed 410-grade stainless steel strips. Each specimen is an ASTM-standard tensile test specimen, with a gauge length of 6 inches (152.4 mm) and specimen thickness of 0.375 inches (9.5 mm). Also a set of Non Linear Ultrasonic (NLU) measurements on two thermally stressed rods was provided. Each of these rods were heated and rapidly cooled with water to incite fatigue at the water application site.

In order to mechanically fatigue the stainless steel specimens, each one is repeatedly subjected to a 2% strain before being released and tested with a series of Barkhausen measurements. Before each tensile test, the specimens are annealed to ensure that all samples have the same initial stress state. The locations of the Barkhausen measurements are generally selected to be symmetric about the center of the specimen. At each location, multiple measurements are typically made to assess repeatability and quantify measurement noise levels. Probe placement is done manually, with the probe lifted away from the surface between successive measurements. The process of stressing the specimen, releasing, and then testing was repeated until the specimen fails.

9.4.1 Analysis

A single test case was provided to the University of Tennessee for evaluation of the feasibility of prognostic indications. This specimen had three sites, and two configurations for each Barkhausen noise measurement. Three samples were taken at each site with both configurations for a total of twelve descriptive measurements taken at each level of strain. The three testing sites are shown in Figure 9.4-1 and are designated as (A) Top, (B) Middle, and (C) Bottom.

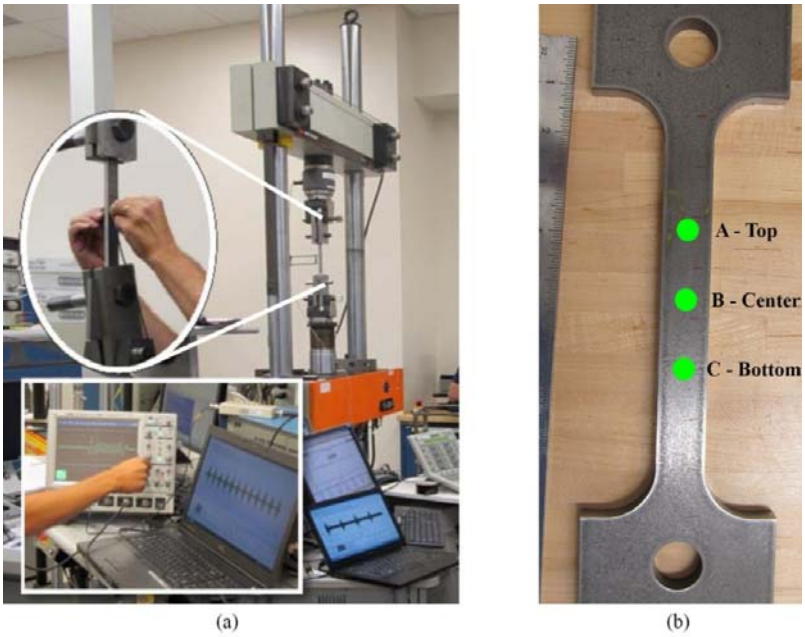


Figure 9.4-1: (a) Experimental setup for tensile test. (b) Typical measurement locations on tensile specimen

The two configurations of Barkhausen noise collection are with the exciters parallel to the direction of strain, and with them perpendicular to the direction of strain, which is shown in Figure 9.4-2.

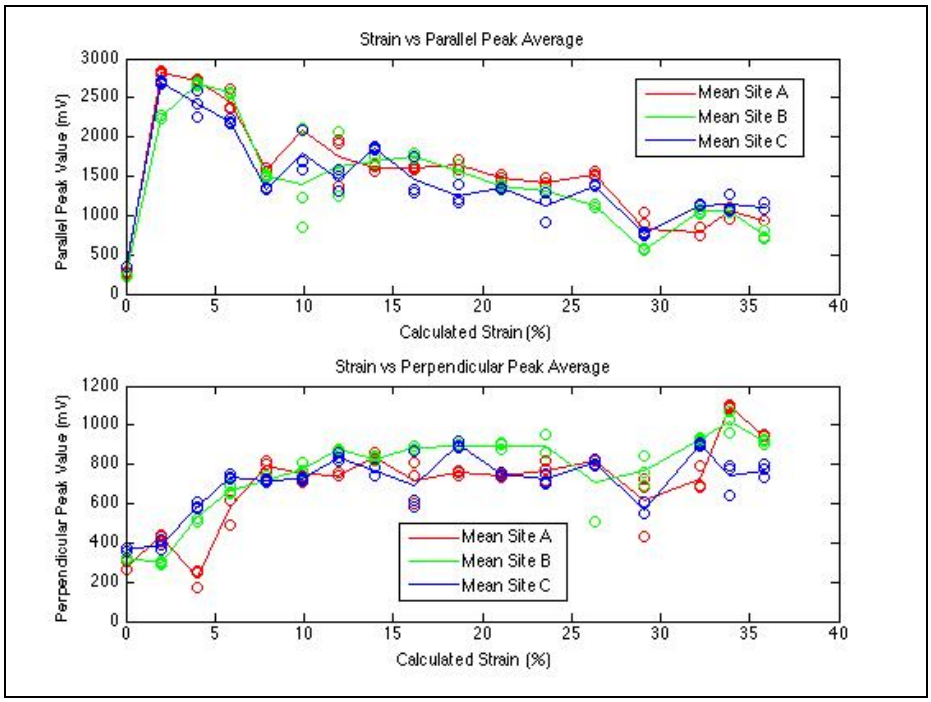


Figure 9.4-2: Barkhausen Peaks vs. Strain

From these plots it is clear that the peak voltage required to initiate Barkhausen noise has strong trends with the strain of the specimen; negative for parallel, and positive for perpendicular. Unfortunately in real systems the direction of strain may not always be known. While one solution to this would be to take measurements in incremental angular directions, this may be time consuming and less than practical in many applications. These charts would also seem to imply that the maximum voltage requirement will be in the direction of strain, and the minimum 90^o shifted from this. As strain increases, the distance between these two values seems to decrease. If there is a continuous progression of these angles, then it can be inferred that the ratio of any two measurements made should show a closing distance of peak voltage with increasing strain. Shown in Figure 9.4-3, two obvious methods for monitoring this would be either the distance between similar 90^o peaks or their ratio or the larger to the smaller.

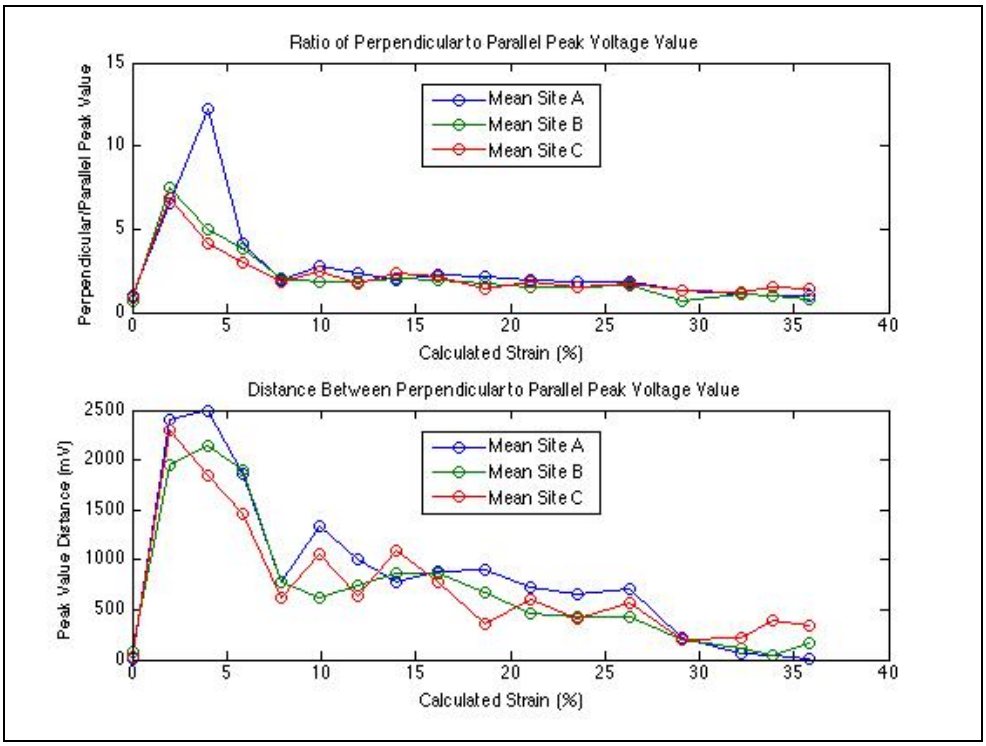


Figure 9.4-3: 90^o Barkhausen Relationships

The figure seems to indicate that the ratio moves to 1 right before failure of the specimen (or distance moves to 0 between opposed directional measurements). The site of most stress (A) has the closest relationship between the two measurement directions, and subsequently is the location of the final failure of the specimen.

Assuming these results are consistent across any arbitrary repeated set of 90° separated samplings at a given location; this shows promise for monitoring and predicting impending failure of passive structures through Barkhausen monitoring. Additional testing in both arbitrary angular configurations and additional sample specimens are required before any major conclusions are drawn, but initial findings are promising.

9.5 Non-Linear Ultrasonic Measurement System

The mechanical strain Non-Linear Ultrasonic (NLU) measurement system test-bed was designed to use two configurations of a "transmit-receive" pair of ultrasonic transducers: through transmission and guided wave. Shown in Figure 9.5-1, are the transmitting and receiving probes of the through-transmission mode, these are on opposite faces of the specimen and aligned center-to-center.

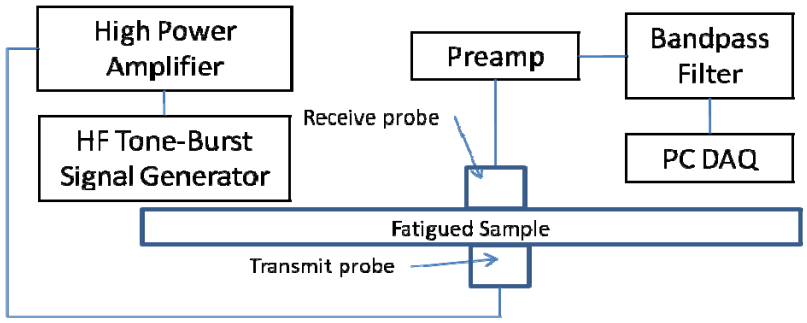


Figure 9.5-1: Schematic of through transmission mode for NLU measurement

Conversely, the guided wave approach recording NLU measurements has the transmitting and receiving transducers mounted on Rexolite wedges separated by a known distance as shown in Figure 9.5-2.

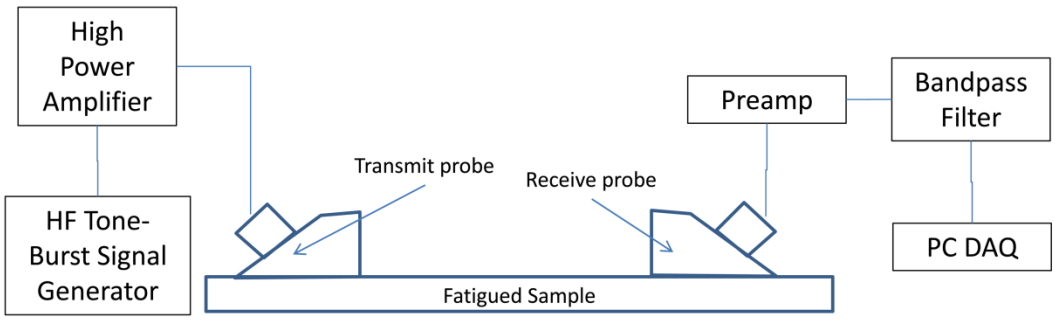


Figure 9.5-2: Schematic of guided wave mode for NLU measurement

In each case, the transmit probe has a center frequency of 5 MHz with a bandwidth of approximately 60%. The receiving transducer has a center frequency of 10 MHz with

approximately the same bandwidth. For each cycled test, a 5-cycle tone burst signal at an incident frequency of 4.5 MHz is transmitted using the transmit transducer, and the resulting response recorded using the receiving transducer. The data is recorded at 500 MHz prior to saving to disk. This process was repeated using tone bursts at 5 MHz and 5.5 MHz. At each frequency, three different input power levels (corresponding to an input voltage to the power amplifier of 100 mV, 200 mV, and 300 mV) were applied to the ultrasonic probe.

9.5.1 Time Series Investigation Results

The University of Tennessee was provided with the raw data files for two separate stainless steel specimens. Sample 1 corresponds to the through-transmission configuration, and Sample 2 corresponds to the guided wave measurements. Working from a data driven standpoint, the initial investigation centered on identifying progressive and trendable features were consistent for both specimens. By first looking at the amplitude of the raw signal on log scale for each of the cycled tests, as in Figure 9.5-3 and Figure 9.5-4, it becomes very clear that there are progressive features in both of the samples as they degrade.

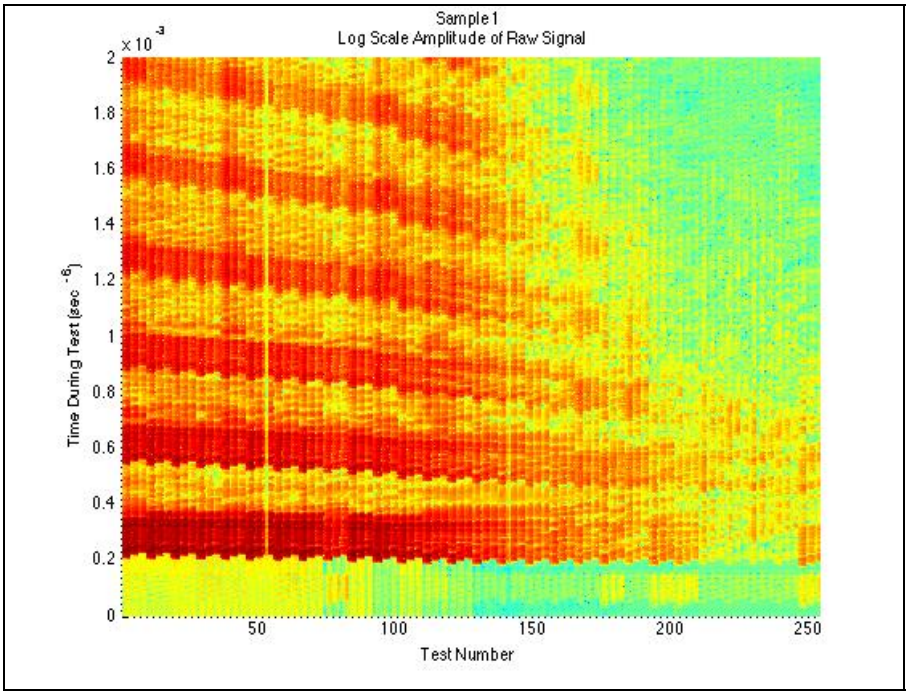


Figure 9.5-3 : Sample 1 Raw Signal Amplitude Though Testing Lifetime

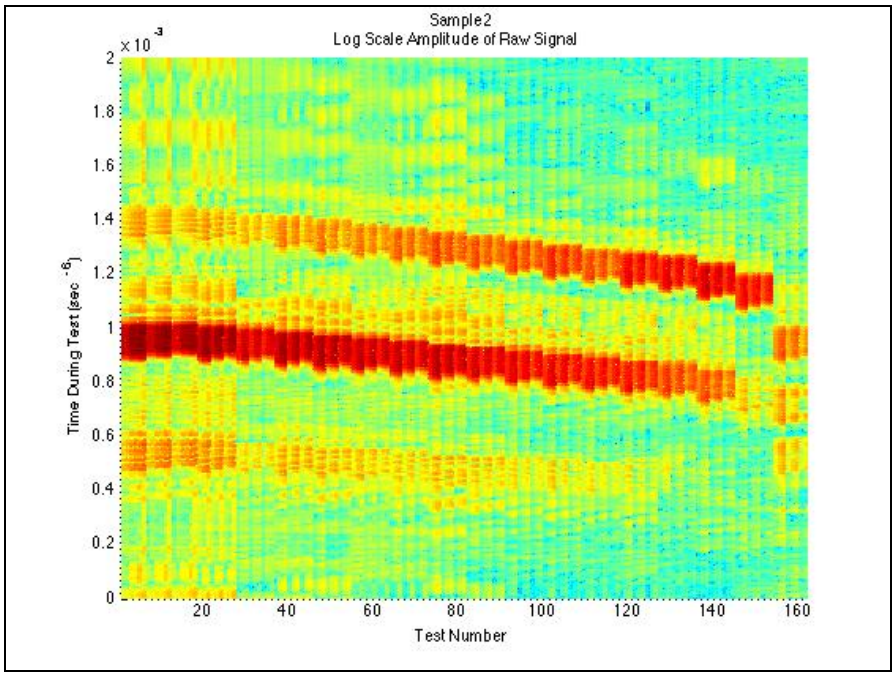
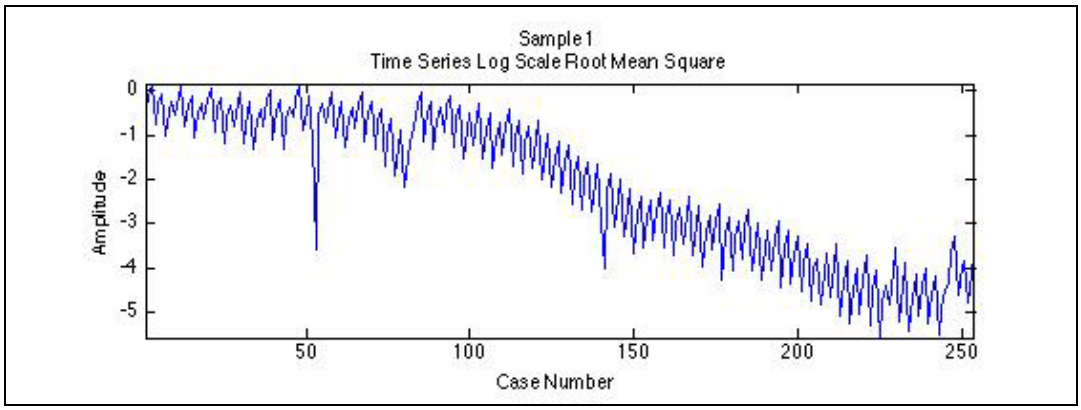


Figure 9.5-4 : Sample 2 Raw Signal Amplitude Though Testing Lifetime

Although the two samples do not exhibit identical progressive features, there are clearly similar evolutions of features between the two samples. The first and most obvious feature is the temporal location of the primary waves shifting with increased cycling and fatigue. However, while Sample 1 (the through-transmission configuration) exhibits significantly more residual wave pulses in different temporal locations than Sample 2 (the guided wave configuration). As an alternative to this temporal tracking that requires user visual inspection and input, or sophisticated peak tracking algorithms, there also appears to be progressive trends in the overall power of the transmitted signal. This can quickly be approximated by the total signal Root Mean Square (RMS) value.



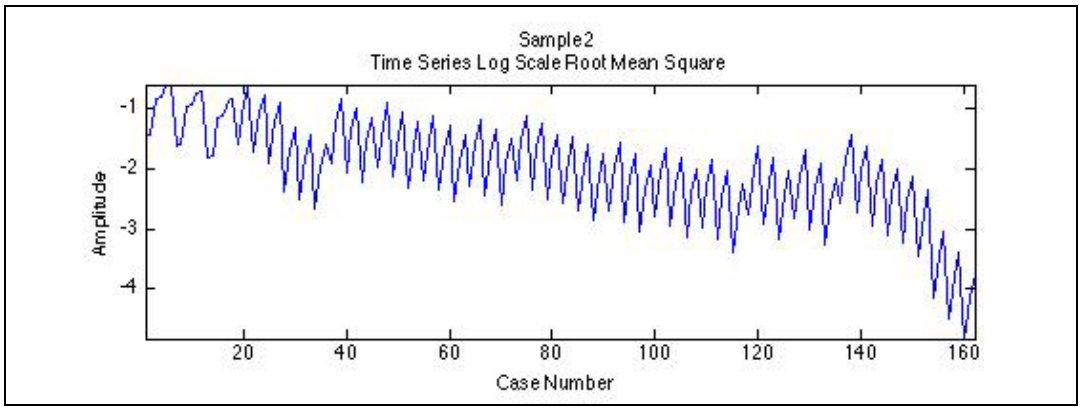


Figure 9.5-5 : Time Series Log Scale RMS over Samples Lifetime

Shown in Figure 9.5-5, the RMS for both samples show progressively downward trends. The "saw-tooth" nature of these signals corresponds to the cyclic nature in the monitoring procedure where the transmissions were made with progressive frequency and power levels for each stress cycle. These can easily be extracted to separate progressive features for a designated transmission setting, as shown in Figure 9.5-6.

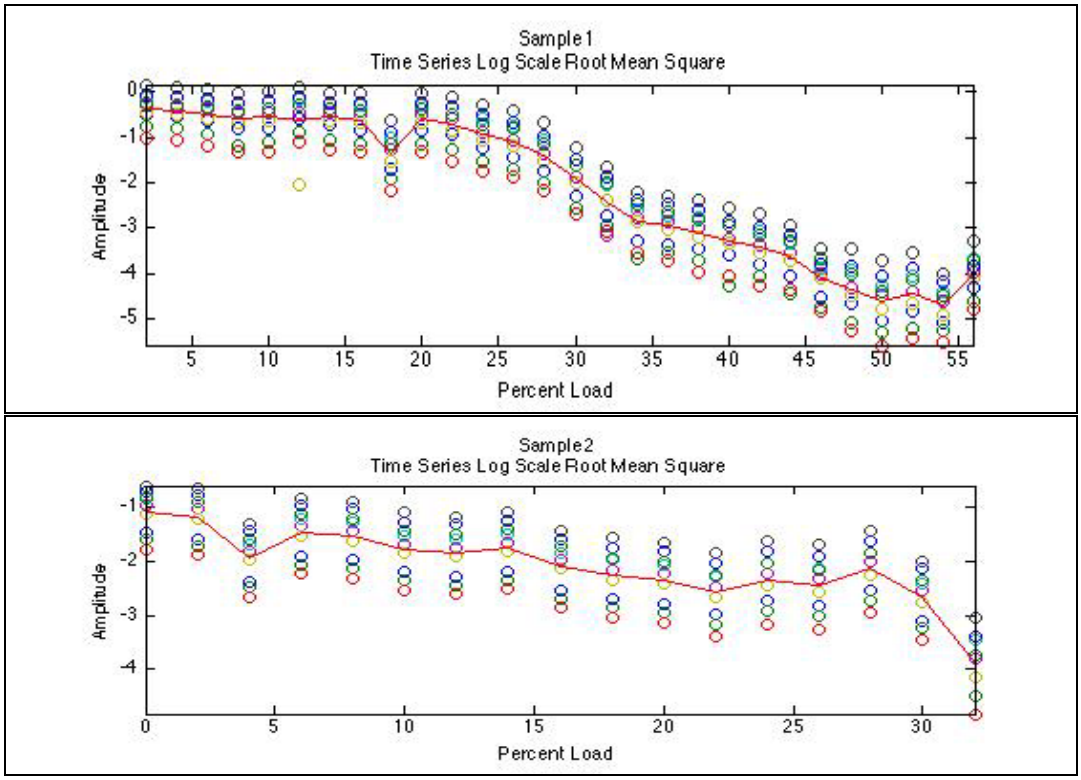


Figure 9.5-6: Time Series Feature vs. Percent Load Separated By Test Configuration

Each of the nine configurations, shown as the colored circles, creates mostly parallel trends through testing with basically no crossover. Both samples exhibit an overall downward average trend, as denoted by the solid line. Additional investigation revealed that the frequency configuration has much greater impact on the transmission received than the voltage level. Regardless, none of these transmissions setting have a significant impact on overall trend, or progressive shape of these features.

9.5.2 Frequency Analysis

Another intuitive area of investigation for any oscillatory signal is the frequency domain, and much like the time domain investigation, even the most cursory investigation reveals progressive trends.

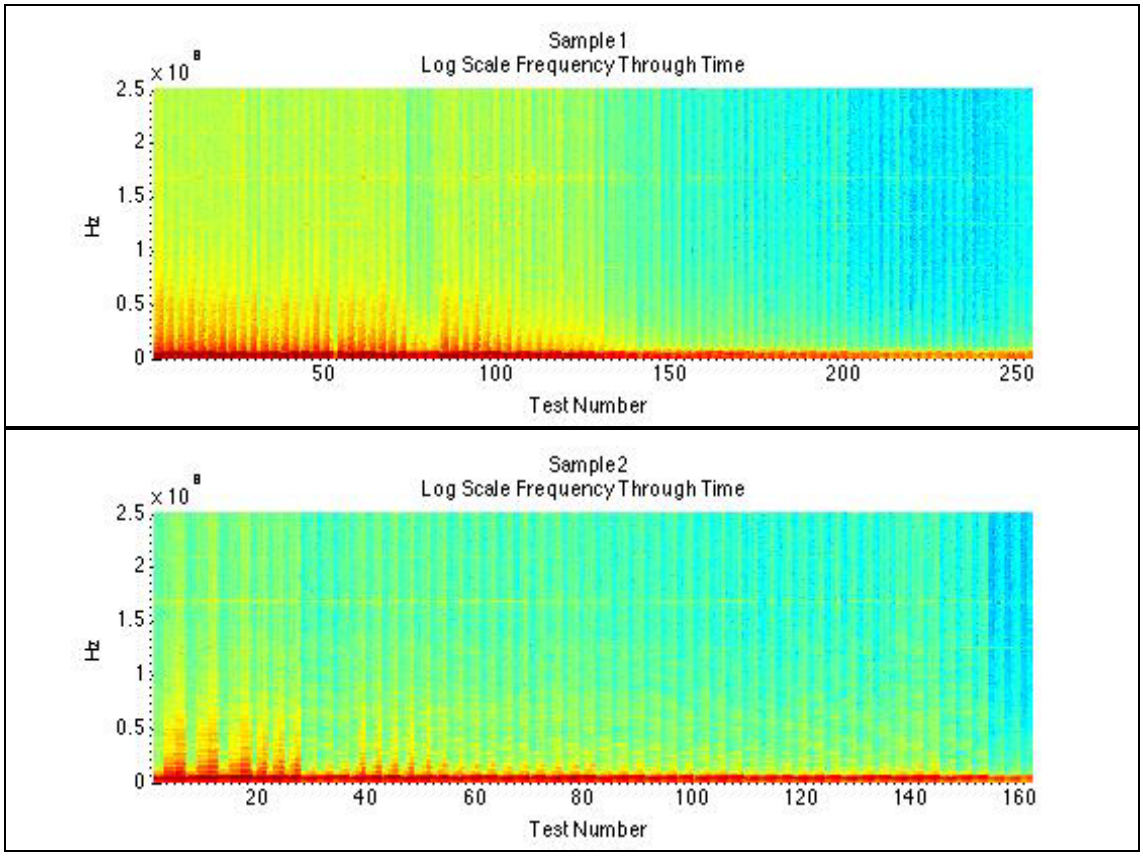


Figure 9.5-7: Frequency Maps Through Sample Lifetime

From Figure 9.5-7, several things become clear. First, although Sample 1 has a much higher range of significant frequencies, both configurations of samples show an overall decreasing amount of power in their dominant frequencies (an echo and conformation of the findings from the time series RMS). Second, both samples have most of their power in the lower frequency

ranges, from approximately 1 to 10MHz. Expanding this range, as in Figure 9.5-8, reveals that several of the dominate frequency ranges experience significant shifts throughout the lifetime of the sample.

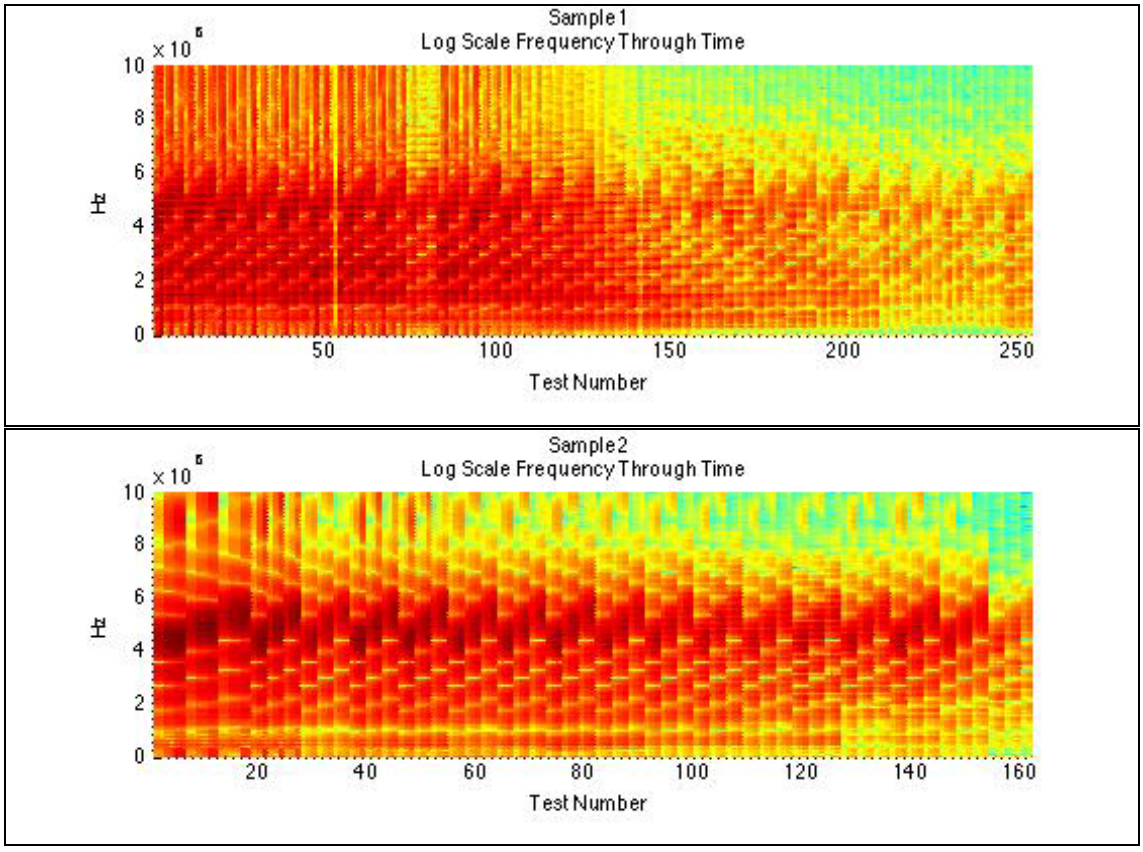


Figure 9.5-8: Expanded Frequency Map through Sample Lifetime

In particular, it becomes obvious that while the dominant frequencies shift locations based on the transmission type and configuration, the loss of power follows degradation and fatigue of the specimen. Major peaks below 10MHz seem to consistently lose power over both specimens. If these frequency power ranges are extracted for both samples, as shown in Figure 9.5-9, a simple feature progression similar to that found with the time series analysis can be seen.

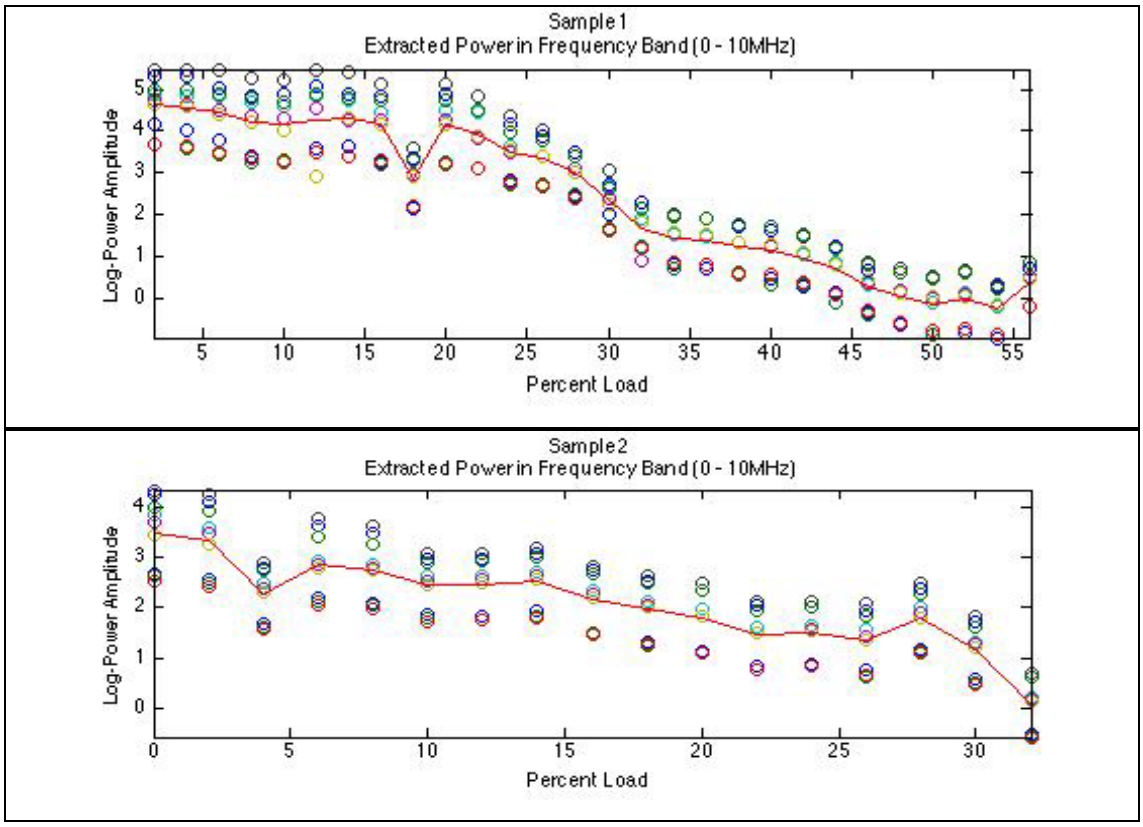


Figure 9.5-9: Extracted Frequency Power over Sample Lifetime

Although the variance from sample configuration is larger than that compared to the time series feature, the overall negative trend remains the same for each sample. In the next section a generic pattern matching method is discussed.

9.5.3 Generic Pattern Matching

Previous work had focused on the generic monitoring of transient signals for deviations from normality with the end goal of prognostic feature extraction [Sharp 2012]. The transmitted pulses in this testing procedure meet all the criteria of transient signals and make a suitable candidate for this process. Assuming nothing about the qualities of the recorded signals, a broad timescale Sharp Transform is applied to the signals and their residuals calculated throughout the lifetime of the sample.

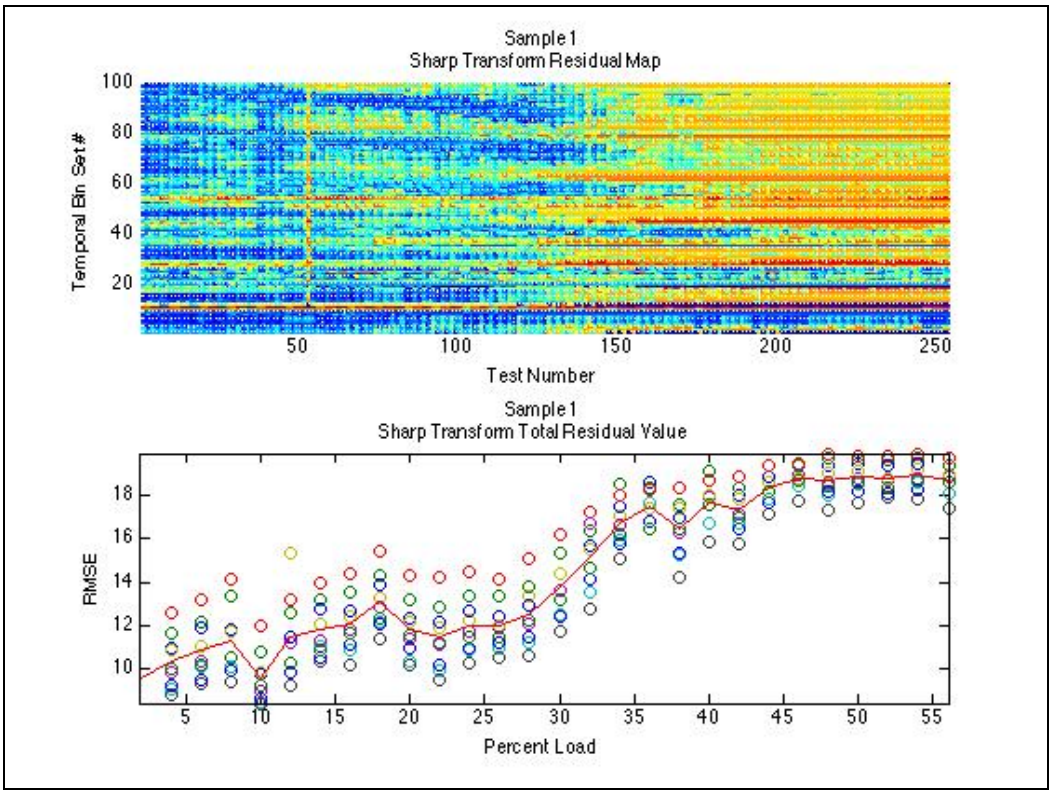


Figure 9.5-10: Sharp Transform Residuals Sample 1

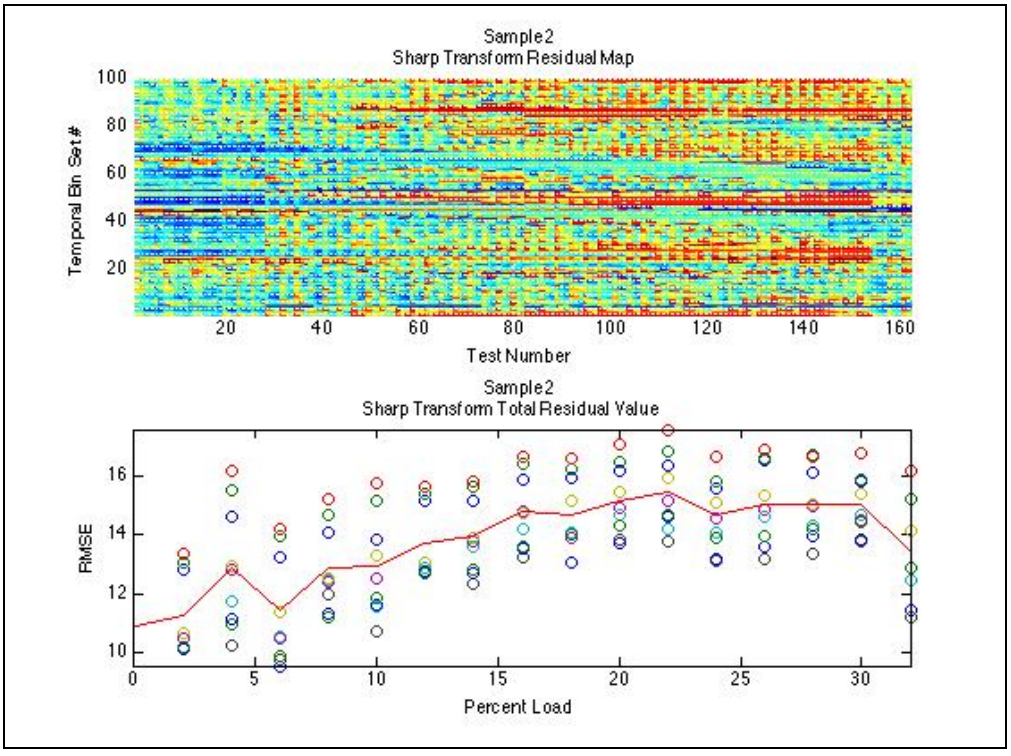


Figure 9.5-11: Sharp Transform Residuals Sample 2

Both samples in the previous two figures show a clear pattern of increased deviation from normality with the increased load. It should also be noted that due to the low number of unique unstressed test configurations, all the various configurations were compared to a single model built upon each corresponding sample. Even so, the progressive trends are clearly evident in both samples.

10. Task 10 and Task 11: Validation of Developed Procedures

After development of the completed lifetime prognostic toolset, a stage of verification and validation is required to evaluate the developed algorithms and tools. What follows is the completion of Task 10 and Task 11 with the procedures for developed algorithms and programs. This section will provide the code and procedures needed to create algorithms using both simulated degradation data and data collected from the physical setups during this project.

10.1 General Transitioning Procedure Example

10.1.1 Type I to Type II

For Type I to Type II transitions, the prior is developed during the Type I analysis. Specifically, the results of Weibull or Gaussian analysis give a RUL distribution with known parameters, which forms the prior. If a simple grouping of historical failures based on similar operating conditions is the Type II analysis, though this method can be generalized for any Type II model that yields a known distribution, the RUL estimate is based off a group of failed components. A fit to the distribution gives the sampling data needed.

Data was simulated, such that 1000 paths were generated following different operating conditions. At greater operating conditions, more stress was added until failure was reached following the LCM. The conditions varied from 1 to 3, with 3 as the harshest operating conditions and most amount of stress. Figure 10.1-1 shows the Type I prior distribution, while Figure 10.1-2 shows the Type II selected distribution.

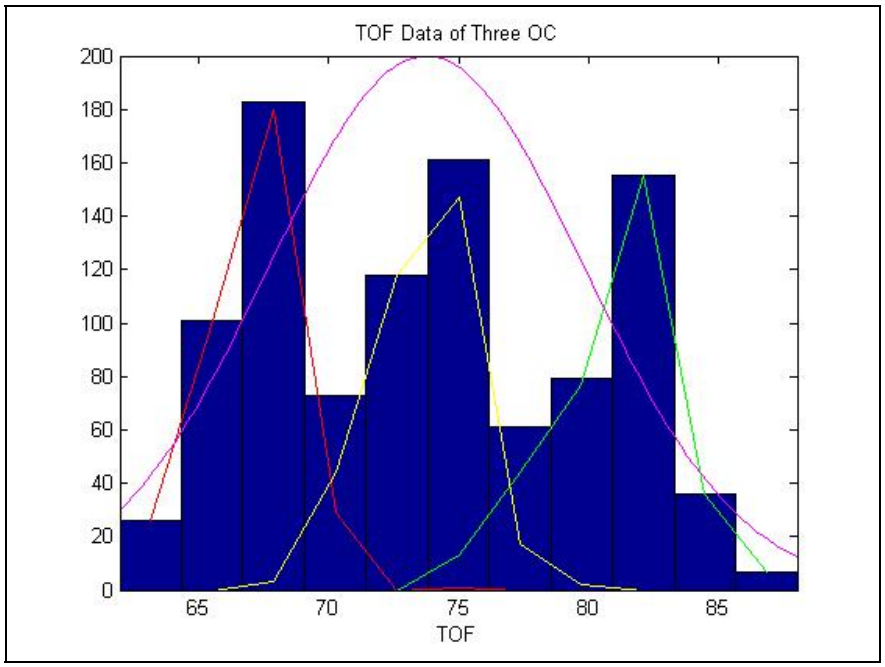


Figure 10.1-1 TOF Distribution for all Historic Data, with Normal Fit

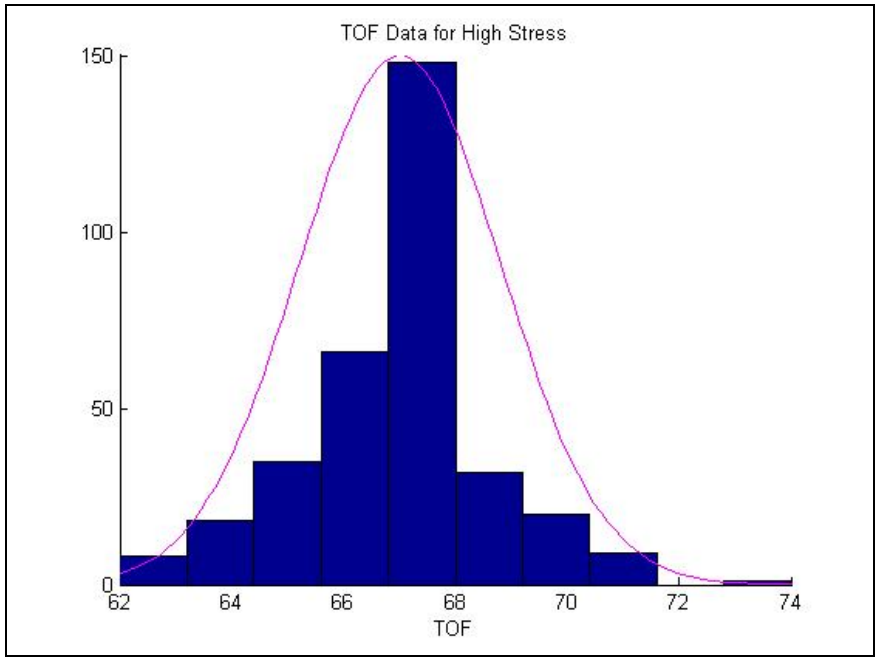


Figure 10.1-2 TOF Distribution for High Stress Paths

Though traditionally a Weibull fit is used for Type I analysis, for simplicity normal distributions were fitted to both data sets. The prior, Type I distribution, was found to be ~N(73.8, 36.4). The distribution of the 337 high stress paths was ~N(67.0, 3.23). Using Equation

3-4 gives an expected value of 67.0 with a posterior variance of .00959. In this case the large high stress sampling group, with small variance, easily swamped out the prior. The posterior mean is very close to the sampling mean, and the posterior variance is very small, meaning there is likely to be very little change in the sampling mean as more data is included.

10.1.2 Transitioning Example Type I/II to GPM Transitions

To demonstrate the various transitions into GPM the 2008 Prognostics Health Management Challenge Data was used. This consisted of 260 training examples of 24 signals ending in failure, and 259 censored testing examples of the same 24 signals. The actual RUL of the testing examples were given for validation.

To apply the GPM the 24 signals were converted into a single prognostics parameter for both datasets. Figure 10.1-3 shows the prognostics parameters for the training set, while Figure 10.1-4 shows the TOF distribution.

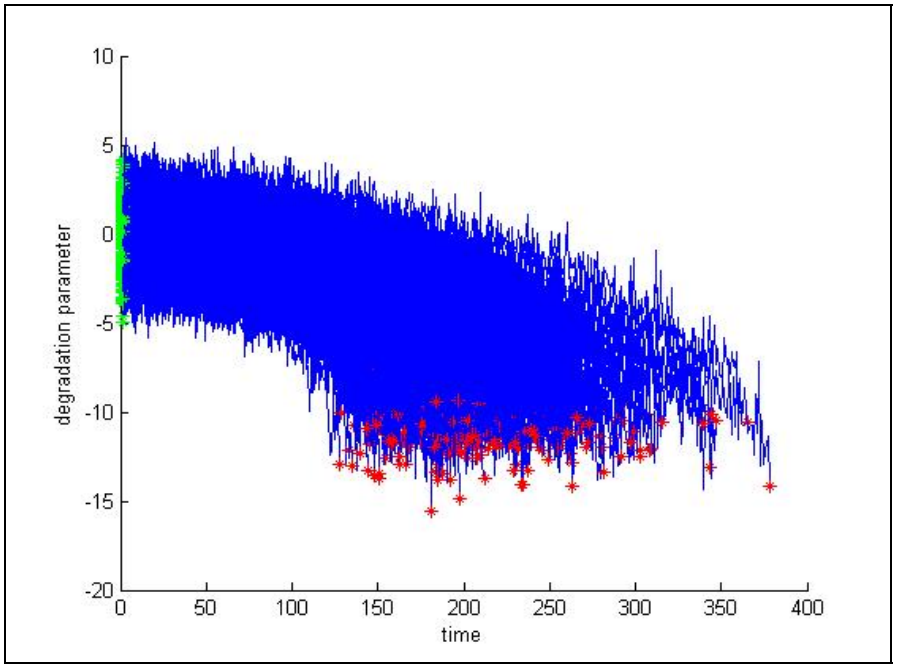


Figure 10.1-3 Prognostics Parameters of Training Set

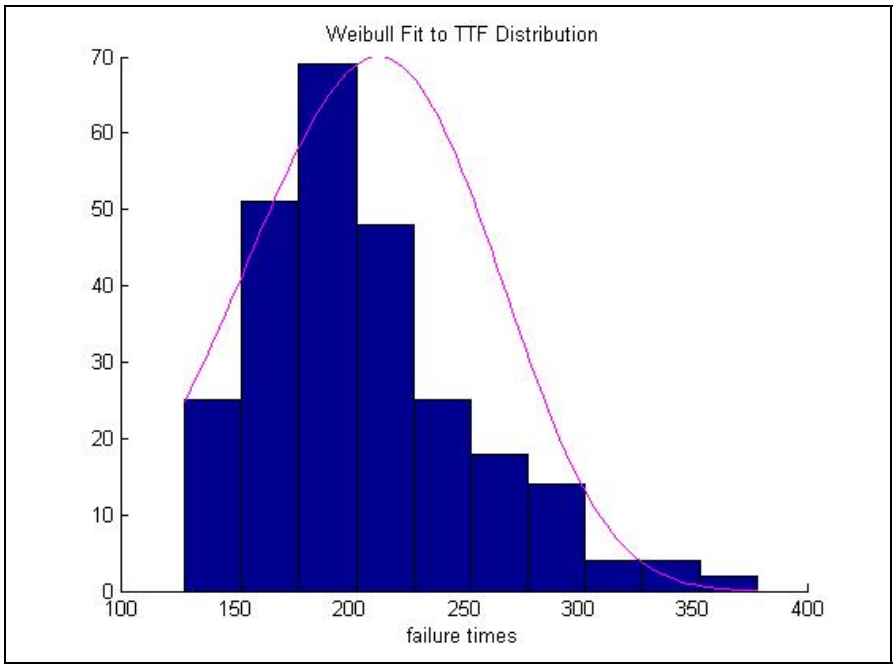


Figure 10.1-4 TOF Distribution with Weibull Fit

Several models were used to accurately estimate the RUL of each of the testing paths. First, for comparison, the standard GPM was applied to each testing path. This was under the assumption that the functional forms of the paths were known to be quadratic. The paths were then extrapolated by using OLS, **Equation 3-7**, to the degradation threshold taken from the failure points of the train paths.

Next the GPM was used with Bayes priors for each of the three parameters. These were obtained from OLS regression of the training set. As stated previously, both the prior and posterior distributions of the path parameters follow the conjugate Gaussian pair. **Equation 3-11** and **Equation 3-12** were applied to obtain the posterior mean and variance, based on the prior distributions:

$$\beta_1 \sim N(-3.87x10^{-4}, 3.05x10^{-8}), \beta_2 \sim N(.0300, 1.19x10^{-4}), \beta_3 \sim N(-.653, 2.20)$$ **Equation 10-1**

The previous two methods are, if not widely established, not new to prognostics (Coble 2010). In order to implement both models, however, extensive knowledge of past failure paths is needed for the functional fit, priors, and critical threshold. In cases of more limited information different models can be applied. First, it is assumed that instead of the full paths, only information about the failure points are known, both the TOFs and critical threshold. This data can also be substituted using information about the process. If there exists a Type I analysis, it can be used to

form the TOF distribution needed. Also if the prognostics parameter has physical significance, and not a complex amalgam of signals, then either an artificial threshold can be established, or it can be set by the physical properties of reaching a certain measured quantity, e.g. a macroscopic crack that cuts across a beam.

However the relevant information is obtained, if the TOF distribution and threshold are known, the previous GPM using OLS regression can be used in a slightly different way. To incorporate Bayes priors, they are treated as additional data points. The variance-covariance matrix Σ contains the information that separates them from regular data points, the prior precision. The difference in scale between data points and the priors gives the Bayes weighting. Using the TOF distribution and threshold, the same concept can be applied. First it must be remembered that one assumption of the OLS model is that errors are only distributed among the response, Y, values. However, the TOF distribution stretches across a different dimension. Classical statistics shows that for two random variables following normal distributions, the sum distribution is also normal, with a variance of the sum of variances [Garcia 2008]. In this case error is a simple linear addition of the X and Y errors. Mathematically:

$$\sigma_t^2 = \sigma_x^2 + \sigma_y^2 \qquad \textbf{Equation 10-2}$$

This total variance is the precision that should be entered into the variance-covariance matrix. For the current data set, while the noise variance is 1.308, the total variance of our prior data point is 2.19e3, much higher than the noise. This means that the prior will be weighed less, as previously seen, it can still be significant in cases in which limited test data is available. The X matrix should be appended with the mean TOF, and the Y with the threshold. Then the GPM can be applied as usual.

Another difficulty in applying the GPM with limited past data is that the functional form may not be known. If a fault is tracked for a process that has not yet been seen, unless extensive knowledge is known about the underlying process and future operating conditions, a path cannot be definitively given a functional form. However most short data sequences can be approximated using linear fits. A strategy of adopting a linear fit for early data points has some benefits. The "y-intercept" parameter constant can easily be obtained, leaving only one unknown parameter, the slope.

For this data set, a Weibull distribution was fit to the "Type I" TOF data. The parameters λ=225.7 and k=4.39 were found. If the slope parameter follows a Gaussian distribution, it can be seen that

the MLE of the slope will correspond to one that will lead the line from its start to the mode of the Weibull at the threshold. The mode can be found:

$$mode = \lambda\left(\frac{k}{k}\right)^{\frac{1}{k}} \; for \; k > 1$$ **Equation 10-3**

And the prior slope mean is:

$$\hat{\beta}_1 = \frac{thresh - \hat{\beta}_2}{mode}$$ **Equation 10-4**

The prior slope variance can be found using Monte Carlo simulations. For the slope to be statistically significant, it can be expected that the 95% confidence interval, or 2 standard deviations, do not encompass 0. Therefore the standard deviation exists between 0 and the mean. Using this logic, 100 values between 0 and the mean were selected as possible standard deviations. For each value, 1000 observations were given following a normal distribution with the mean and test deviations. The standard deviation that gave a distribution with the closest Euclidean distance to the Weibull fit was chosen as the prior slope deviation. Using the prior slope mean, variance, intercept mean, and intercept variance, which is used as the noise variance, can facilitate in forming the prior distributions for a linear model. It can be expected that because it is known that this data set follows more closely to a quadratic form, rather than linear, there will be some error.

To summarize, four different prediction methods were applied to the test set to estimate RUL for each case: (1) OLS regression, (2) OLS with Bayesian prior parameters, (3) using Type I data as a data point, and (4) using Type I data for a linear path to get Bayesian prior parameters.

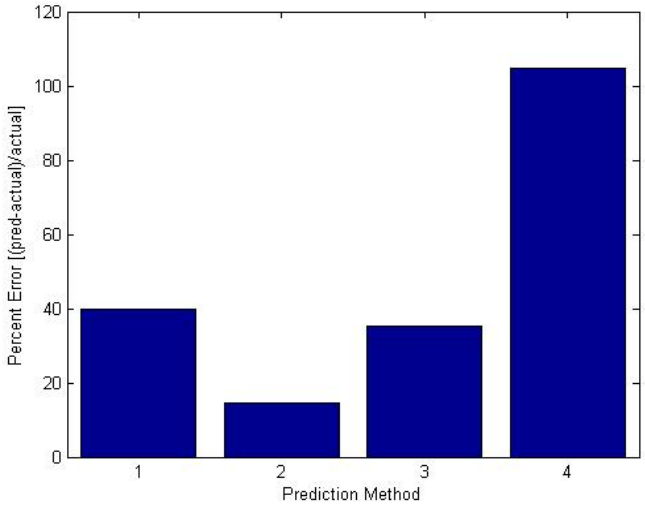


Figure 10.1-5 Comparison of OLS Models for Predicting RULs on PHM Data

As expected, Figure 10.1-5 shows the method that minimizes the error is (2). This model takes advantage of the most data, in the form of the average parameter distributions from previously failed cases. Model (3) restricts this data greatly to reduce the error from the standard GPM (1) model. Though models (3) and (4) take advantage of the same data, (4) forces the GPM to follow a linear path, though we know this to not be the case.

10.2 Pump Impeller Example Case

10.2.1 Type I Prognostic Model Development

The total number of tests that were performed in the impeller degradation experiment was 28, but 10 of these were removed because they had unsuitable features for model development. To run the LOOCV code for the Type I models for a certain percentage of system life is accomplished by:

```
pLife = [0.1:0.1:0.9]';  % percent of life %
% Type I Prognostics Model
RUL = nan(size(mod_dat,1),numel(pLife));
RUL_act = nan(size(mod_dat,1),numel(pLife));
TOF_act = nan(size(mod_dat,1),numel(pLife));

 for ii = 1:size(mod_dat,1)
    train = mod_dat(:,2); train(ii,:) = [];
    test = mod_dat(ii,2);
    model = initTypeI(train); %,'distribution','normal');
    % find the total lifetime of test case %
    TOF = test;
    currsys = [pLife*TOF];
    a = runTypeI(model,currsys);
    RUL(ii,:) = a.RUL;
```

```
    RUL_act(ii,:) = TOF - currsys(:,1)';
    TOF_act(ii,:) = TOF*ones(1,9);
end

MAPEI = mean(abs(RUL-RUL_act)./TOF_act)*100;
MEDIAPEI = median(abs(RUL-RUL_act)./TOF_act)*100;
figure; plot(pLife,MEDIAPEI)
legend('Type I')
xlabel('Percent of Life')
ylabel('MAPE')
title('MAPE for Type I LOOCV Models')
figure; plot(ones(18,1)*pLife',abs(RUL-RUL_act)./TOF_act*100,'.')
hold on
plot(pLife,100*ones(size(pLife)),'k--')
xlabel('Percent of Life')
ylabel('APE')
title('APE for Type I LOOCV Models')
```

This code will perform the LOOCV method, calculate the ToF error and plot the results.

10.2.2 Type II Prognostic Model Development

The Type II model developed for this research was the Proportional Hazards Model (PHM), which uses the operating conditions that the average component experiences during normal operations as additional information for RUL estimates. This new type of model requires four pieces of information that are placed into a specific structure, this information is:

- Operating Conditions- this are the operating used in testing, here 100, 75 and 50 percent outlet line open
- Failure times – these must be matched to each operating condition
- Frequency of Failure – must also me matched to operating conditions, if there were 5 failures that occurred at 3 days for the 100% operating condition, then the frequency value for 3 days is 5
- Censoring – tests that should be included or excluded from the model, 0 for no censoring and 1 for censored

This information is placed into an **n** x 4 matrix; in this case there were 15 different combinations of operating conditions and failure times so the matrix for the development of the PHM is size 18 x 4.

The data structure for the PHM looks like:

```
100.0000    1.8000    1.0000         0
```

100.0000	4.9000	1.0000	1.0000
100.0000	4.3000	1.0000	0
100.0000	3.7000	1.0000	1.0000
100.0000	2.9000	1.0000	1.0000
100.0000	0.4500	1.0000	1.0000
75.0000	0.4700	1.0000	0
75.0000	0.3500	1.0000	0
75.0000	0.7700	1.0000	0
75.0000	33.0000	1.0000	0
75.0000	5.8000	1.0000	1.0000
75.0000	2.4000	1.0000	1.0000
75.0000	5.9800	1.0000	0
75.0000	1.1900	1.0000	1.0000
75.0000	5.9700	1.0000	0
50.0000	10.0400	1.0000	0
50.0000	24.7200	1.0000	0
50.0000	5.1900	1.0000	0

Once this data is place into a matrix, then the development of the PHM can begin. The first step is to generate the reliability functions for each of the operating conditions and then check for proportionality. The proportionality is checked by taking the log of the negative log of the reliability functions and then plotting the results. Next the code that will generate the results and plots for this first step is shown.

```
%PHM Code
%seperate data based on operating condition
d1 = mod_dat(1:5,:);
d2 = mod_dat(6:15,:);
d3 = mod_dat(16:18,:);

%weibull fit on the three operating conditions
dc=[0:70];
phat1=wblfit(d1(:,2),5,d1(:,4),d1(:,3));
phat2=wblfit(d2(:,2),5,d2(:,4),d2(:,3));
phat3=wblfit(d3(:,2),5,d3(:,4),d3(:,3));
```

```
%generate and plot reliability functions
OC1=wblcdf(dc,phat1(1),phat1(2));
plot(1-OC1,'r');hold on
OC2=wblcdf(dc,phat2(1),phat2(2));
plot(1-OC2,'b');hold on
OC3=wblcdf(dc,phat3(1),phat3(2));
plot(1-OC3,'k');hold off
legend('100% Open','75% Open','50% Open')
ylabel('Fractional Likelihood of Failure')
xlabel('Failure Time (days)')
title('Type II Reliability Functions')

%calculate and plot proportionality of reliability functions
oc11=-log(1-OC1);
oc22=-log(1-OC2);
oc33=-log(1-OC3);

figure;plot(log(oc11),'r');hold on
plot(log(oc22),'b');
plot(log(oc33),'k');hold off
legend('100% Open','75% Open','50% Open')
ylabel('log of Fractional Likelihood of Failure')
xlabel('Failure Time (days)')
title('Proportionality Check of Reliability Functions')
```

The first plot generated is the reliability functions for each operating condition and the second is the proportionality check of the reliability functions, shown in Figure 10.2-1 .

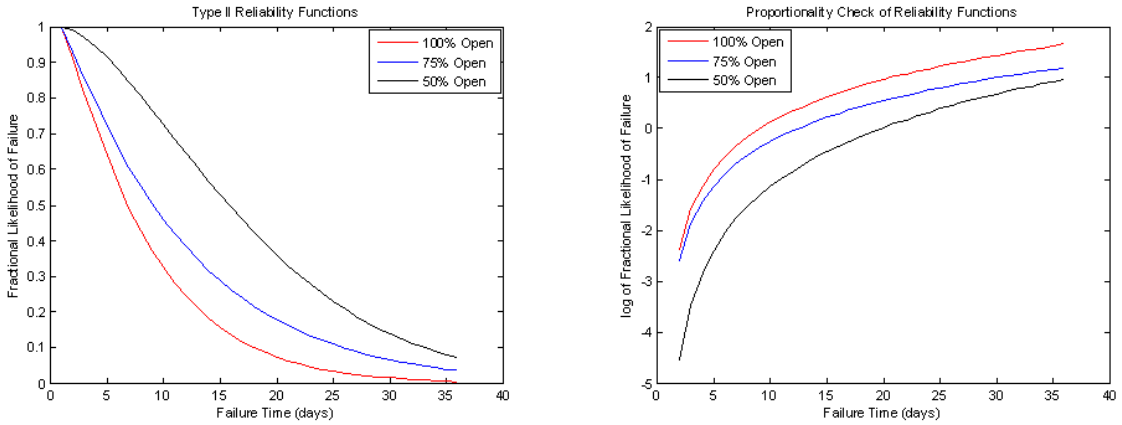


Figure 10.2-1: Operating Condition Reliability Functions & Proportionality Check

Next, initialize the Type II model and run using the current system times to obtain RUL estimates and errors as in the Type I model:

```
%% Type II
 % initialize RUL %
RUL = nan(size(mod_dat,1),numel(pLife));
RUL_act = nan(size(mod_dat,1),numel(pLife));

for ii = 1:size(mod_dat,1)
    train = mod_dat; train(ii,:) = [];
    test = mod_dat(ii,:);
    model = initPHM(train(:,1),train(:,2),'baseline',100);
    % find the total lifetime of test case %
    TOF = test(2);
    test_cov = test(1);
    currsys = [pLife*TOF test_cov*ones(size(pLife))];
    RUL(ii,:) = runPHM2(model,currsys);
    RUL_act(ii,:) = TOF - currsys(:,1)';
end


MAPEII = mean(abs(RUL-RUL_act)./TOF_act)*100;
MEDIAPEII = median(abs(RUL-RUL_act)./TOF_act)*100;
```

The same plotting functions can be used as in the Type I model.

10.2.3 Type III Prognostic Model Development

The Type III model used for this research was the General Path Model (GPM), which requires degradation paths that are extracted as features from the raw data. The majority of the work in the GPM development is the extraction of features that show trends seen in the test cases. An example of a trend is a signal mean value steadily increasing or decreasing over time. Several feature extraction methods were explored during the course of research, including statistical measures, peak tracking in the frequency domain, and failure residuals generated from an AAKR model. For brevity, only the generation of the model failure residuals will be shown as a feature extraction technique.

The data was collected as text files in LabView and then transformed into data matrices in MATLAB. Each pump has three measured signals: vibration, differential pressure and current. Each pump test is labeled in sets based on the operating conditions used; an example of the file "1set_100_finaldata" has three pump tests is seen by typing "whos" in command line once the data file is loaded into the workspace:

```
 whos

  Name           Size                Bytes  Class

  pump1set1      2519040x3           60456960  double
```

pump2set1	1176000x3	28224000	double
pump4set1	2519040x3	60456960	double

In total, there were 7 sets of pump tests, with a total of 24 pumps tested to failure. An OLS method was used that combined the time domain features into a prognostic parameter. As in the previous models, the LOOCV is the same for the GPM:

```
%% Type III
clear currsys

% initialize RUL %
RUL = nan(size(new_failtimes'),numel(pLife));
RUL_act = nan(size(new_failtimes'),numel(pLife));

for ii = 1:size(new_failtimes')
    train = cut_progparams; train(ii) = [];
    test = cut_progparams{ii};
    model = initGPM(train,'npop',0);
    % find the total lifetime of test case %
    TOF = size(test,1);
    t = floor(pLife*TOF);
    for jj = 1:numel(t)
        currsys{jj} = [[1:t(jj)]' test(1:t(jj),:)];
    end
    a = runGPM(model,currsys);
    RUL(ii,:) = a.RUL*10/(24*60);
    RUL_act(ii,:) = (TOF - t')*10/(24*60);
end

MAPEIII = mean(abs(RUL-RUL_act)./TOF_act)*100;
MEDIAPEIII = median(abs(RUL-RUL_act)./TOF_act)*100;
```

The plotting functions can again be applied so that the error results may be compared.

10.3 Monitoring and Prognostics Tutorial

The following example employs the PEM and PEP Toolboxes to develop monitoring, fault detection, and prognostic models for a simulated turbofan engine. The data and a description of its simulation are available at http://ti.arc.nasa.gov/project/prognostic-data-repository. The necessary data file is included in the PEP Toolbox. The code below can be pasted into MATLAB; results are shown after the generating code.

The PHM Challenge data set consists of 218 cases of multivariate data that track from nominal operation through fault onset to system failure. Data were provided which modeled the damage propagation of aircraft gas turbine engines using the Commercial Modular Aero-Propulsion

System Simulation (C-MAPSS). This engine simulator allows faults to be injected in any of the five rotating components and gives output responses for 58 sensed engine variables. The PHM Challenge data set included 21 of these 58 output variables as well as three operating condition indicators, for a total of 24 measured variables. Each simulated engine was given some initial level of wear, which would be considered within normal limits, and faults were initiated at some random time during the simulation. Fault propagation was assumed to evolve in an exponential way based on common fault propagation models and the results seen in practice. Engine health was determined as the minimum health margin of the rotating equipment, where the health margin was a function of efficiency and flow for that particular component; when this health indicator reached zero, the simulated engine was considered failed.

This example application of health monitoring will look at developing a system without the benefit of the domain knowledge described above. Here, the available data is used to develop each module from sensor selection through modeling, fault detection, and prognosis.

```
% Script to run through model development and optimization, system        %
% monitoring, fault detection, and prognostic model development with the  %
% PEM and PEP Toolboxes                                                   %

% System will be developed for the 2008 PHM Challenge Data, available at  %
% http://ti.arc.nasa.gov/project/prognostic-data-repository               %

load PHMchalldata

%% Divide Data

% Assume first 15% of each data run is fault free for monitoring system   %
% development and optimization.                                           %

train = [];
for i = 1:260
    train = [train;trn{i}(1:floor(0.15*size(trn{i},1)),:)];
end

% Divide "fault free" data into training, test, and validation data       %
[x1 x2 x3 x4 x5 x6 x7 x8] = vensample(train,8);
training = [x1; x3; x5; x7];
testing = [x2;x6];
validation = [x4;x8];

clear x1 x2 x3 x4 x5 x6 x7 x8

%% Choose Monitoring Model Inputs
```

```
% Model inputs are chosen based on linear correlations between available  %
% variables.  Correlations with magnitude less than 0.3 are considered    %
% unuseful; those with magnitude between 0.3 and 0.7 may have some        %
% predictive power, and those with magnitude greater than 0.7 are         %
% considered good predictors.                                             %

% The PEM function AAGROUP() divides data into groups for auto-associative%
% model development.  It uses a cut-off of 0.7 to identify appropriate    %
% groups from the data.  This function gives us two groups, as shown.     %
% Notice that some variables are members of both groups.                  %

Groups = aagroup(training);

disp(['Group 1 :',10])
disp(Groups{1})
disp(['Group 2 :',10])
disp(Groups{2})
gcplot(training,Groups)
```

```
Variables:24
Observations:3972
Removed Sensors:
The number of the removed sensors:0
Group 1 :

  Columns 1 through 20

     1     2     4     5     6     7     8     9    10    11    12    13    14
15    17    18    19    20    21    23

  Column 21

    24

Group 2 :

     3    11    14    16    17    18    21    22
```

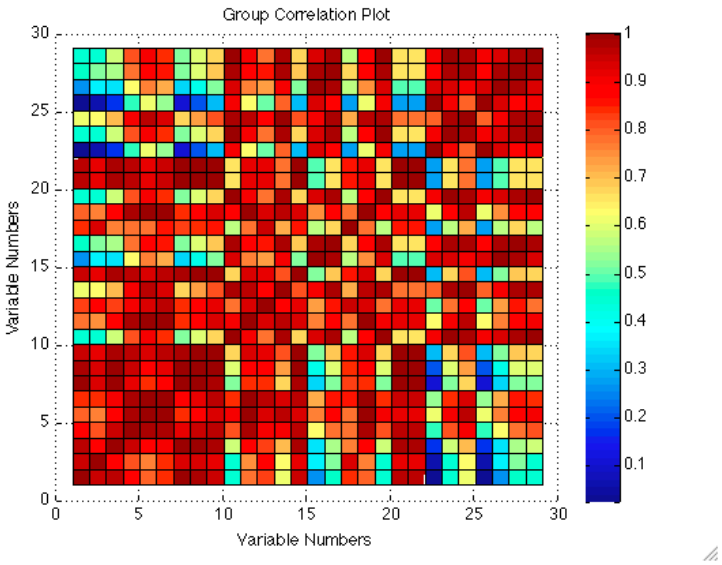


Figure 10.3-1: Correlation Coefficients Between Signals

```
% We can use the variables in Group 1 to make a monitoring system. This    %
% model will have 21 variables.                                             %

train = training(:,Groups{1});
test = testing(:,Groups{1});
val = validation(:,Groups{1});

% Initialize Monitoring System Model %
model = initmodel('aakr',train,'nmem',500);
model =
setmsa(model,'plotresults',0,'fdetmethod','sprtn','variablenames',num2cell(Group
s{1}));
model = optmodel(model,test,'error','bandwidth',[0.5 0.75 1.0 1.5]);
model = modchar(model,val);

%% Monitoring and Fault Detection

% Now we can use our models for system monitoring, residual generation and%
% fault detection.                                                          %

% Extract SPRT attributes for use in fault detection                        %
m1 = model.attributes.error.mean;
v1 = model.attributes.error.std.^2;
t1 = model.attributes.sprttolerance;

% Calculate model predictions, residuals, and fault hypotheses for each    %
% model                                                                     %

trn2 = cell(size(trn));
```

```
Fhyp = cell(size(trn));

for i = 1:length(trn)
    trn2{i} = trn{i}(:,Groups{1});
end

res = residgen(model,trn2);

for i = 1:260
    Fhyp{i} = sprtn(m1,v1,res{i},0.01,0.1,t1);
end

%%

% Let's look at the results for run #2                                    %
for j = [2 5 20]
    figure
    subplot(2,1,1); plot(Fhyp{2}(:,j),'o');
    ylabel('Fault Hyp');
    title({['Variable ' num2str(model.attributes.variablenames{j})],...
        'SPRT Fault Hypotheses'});
    axis([-inf inf -0.05 1.05])
    subplot(2,1,2);plot(res{2}(:,j));
    xlabel('Time (cycles)');
    ylabel('Error');
    title('Residuals');
    axis([-inf inf -inf inf])
end
```

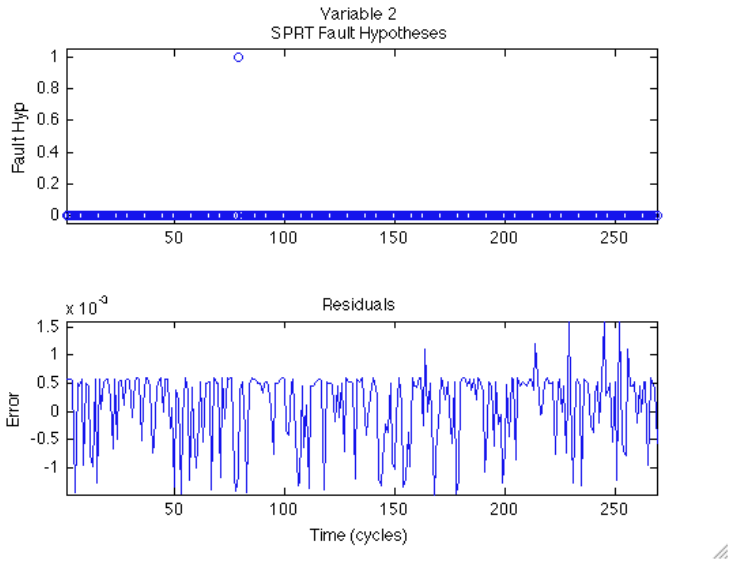


Figure 10.3-2: Sequential Probability Ratio Test of Variable 2

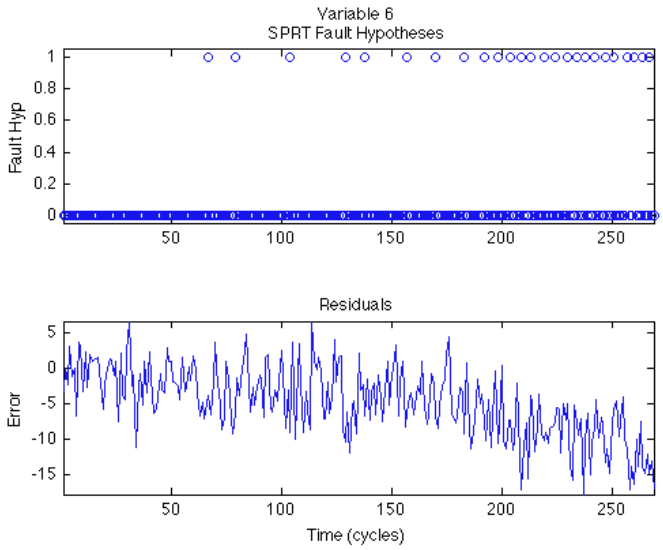


Figure 10.3-3: SPRT of Variable 6

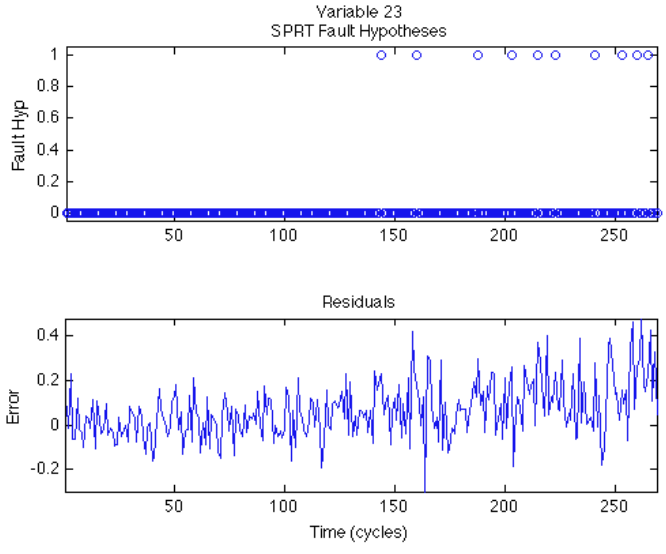


Figure 10.3-4: SPRT of Variable 23

```
%% Prognostic Models - Type I Model

% Type I models are traditional time-to-failure models. The most common   %
% distribution for developing this kind of model is the Weibull           %
% distribution, which can model burn in, random failure, and wear out. The%
% Time of Failure for each failed case is the only data needed to develop %
% this type of model.                                                     %

TOF = zeros(length(trn),1);
for i = 1:length(trn)
```

```
    TOF(i) = length(trn{i});
end

typeI = initprog('typeI',TOF)
typeI.parameters

typeI =

            type: 'TypeI'
    distribution: 'weibull'
      parameters: [1x1 struct]
            data: [1x1 struct]


ans =

     beta: 4.3883
    theta: 225.6644

% The RUL of a new system is estimated based only on the amount of time   %
% that system has been running.                                           %

current_time = zeros(length(tst),1);
for i = 1:length(tst)
    current_time(i) = length(tst{i});
end
TypeIOut = runtypeI(typeI,current_time);
RUL_typeI = TypeIOut.RUL;

MAPE_typeI = mean(abs(RUL - RUL_typeI)./RUL*100);

figure; hold on
plot(RUL,RUL_typeI,'bo');
plot(0:200,0:200,'r--');
hold off
box
xlabel('Actual RUL (cycles)')
ylabel('Estimated RUL (cycles)')
title(['Type I RUL Estimation Error : MAPE = ' num2str(MAPE_typeI)])
```

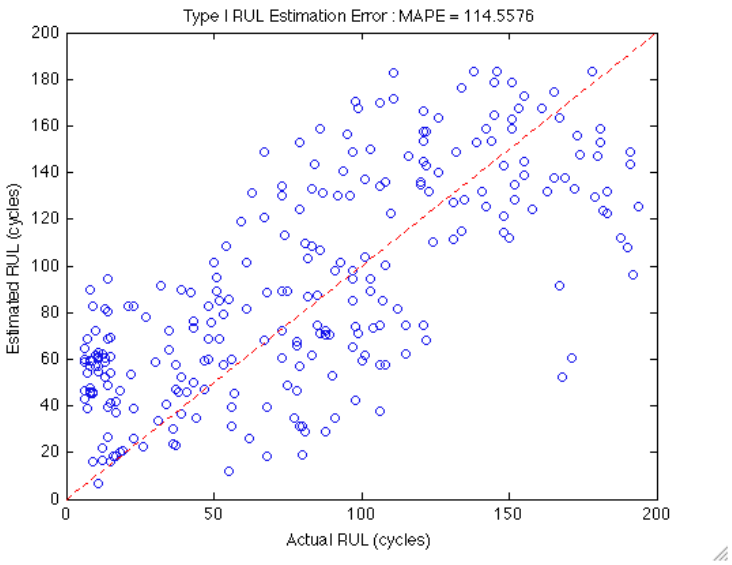


Figure 10.3-5: Type I RUL Estimation Error

```
%% Prognostic Models - Type II Model

% Type II models consider the past and expected future operating          %
% conditions of a system when making prognostic estimates.  One example of%
% a Type II algorithm is the Markov Chain model. This model uses a        %
% transition probability matrix to simulate possible future operating     %
% conditions and relates these conditions to a degradation measure. The   %
% PHM Challenge Data has six distinct operating condtions, so it may be   %
% well suited to this type of model.                                      %

% Format historic operation condition progressions to be numbered 1 - 6.  %
% These numbers have no ordinal relationship.                             %

old_oc = cell(size(trn));
for i = 1:length(trn)
    old_oc{i} = trn{i}(:,1);
end

[new_oc map] = MCdata(old_oc);

typeII = initprog('MC',new_oc,'RULcon',0.5)
typeII.Q
typeII.b

typeII =

         type: 'MC'
            Q: [6x6 double]
            u: [0.1577 0.1346 0.1231 0.1885 0.1385 0.2577]
```

```
            f: @(b,t)t*b
            b: [6x1 double]
    threshold: 100
         npop: 1000
       RULcon: 0.5000


typeII.Q =

    0.1474    0.1520    0.1512    0.1521    0.1430    0.2544
    0.1440    0.1508    0.1502    0.1472    0.1524    0.2554
    0.1455    0.1466    0.1517    0.1468    0.1588    0.2506
    0.1543    0.1568    0.1479    0.1507    0.1415    0.2488
    0.1502    0.1551    0.1542    0.1465    0.1511    0.2431
    0.1536    0.1460    0.1518    0.1488    0.1500    0.2498


typeII.b =

    0.6506
    0.3333
    0.4624
    0.5269
    0.7096
    0.2334

% Format test path operating conditions to the MC classes 1 - 6 using the %
% map identified previously.                                               %

test_oc = cell(1,259);
for i = 1:259
    test_oc{i} = tst{i}(:,1);
end
test_oc = MCdata(test_oc,'map',map);

% Estimate the 50% RUL for each test run.                                  %

TypeIIOut = runMC(typeII,test_oc);
RUL_typeII = TypeIIOut.RUL;

%%

MAPE_typeII = mean(abs(RUL - RUL_typeII)./RUL*100);

figure; hold on
plot(RUL,RUL_typeII,'bo');
plot(0:200,0:200,'r--');
hold off
```

```
box
ylabel('Estimated RUL (cycles)')
xlabel('Actual RUL (cycles)')
title(['Type II RUL Estimation Error : MAPE = ' num2str(MAPE_typeII)])
```

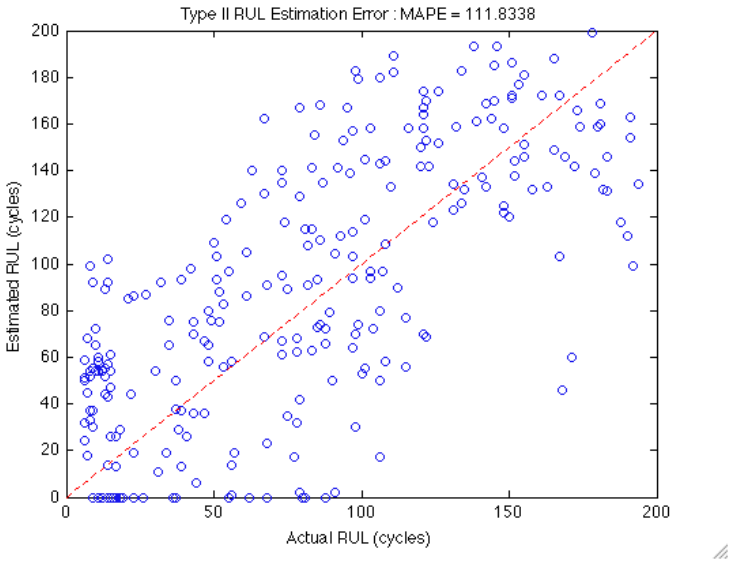


Figure 10.3-6: Type II RUL Estimation Error

```
% We see that the results of this model are not very good. If we look back%
% at the transition probability matrix and linear fit, we see that we do  %
% not get much extra information from the markov chain formalism. The     %
% probability of transitioning between any two states is nearly equivalent%
% and the total time spent in any one operating condition does not seem to%
% give us much information about the degradation level.                   %


%% Type II Markov Chain with Bayes


% It is possible to combine the Type II Markov Chain model with the Type I
% model using Bayesian transition. This type of transition however is not
% recommended when using redundant data for both models, as in this case.
% If however the sources of data are different, then the models will
% combine information from both sources to give a more precise RUL
% estimate.

TypeIIBOut = runMC(typeII,test_oc,typeI);
RUL_typeII_B = TypeIIBOut.RUL;

MAPE_typeII_B = mean(abs(RUL - RUL_typeII_B)./RUL*100);
```

```
%% Prognostic Models - Type III Model

% Type III models consider the actual condition of the system, either     %
% measured or inferred from other measurements.  These condition          %
% measurements are fit to a parametric model which is then extrapolated to%
% a pre-defined critical failure threshold.                               %

% The first step is to identify an appropriate prognostic parameter.      %
% Monitoring model residuals are a natural choice for prognostic          %
% parameters because they naturally characterize how "far" the system is  %
% operating from normal behavior. For this example, we are looking for the%
% optimal linear combination of the 21 residuals based on a sum of the    %
% three suitability metrics: monotonicity, prognosability, and            %
% trendability.                                                           %


% The optimized parameter includes a subset of the monitoring system      %
% residuals. Residuals are chosen for inclusion in the GA by calculating  %
% the fitness of each individual residual and including only those with   %
% total suitability over 2.0                                              %
param = optparam(res,'inputs','subset','cutoff',1.0)
par = paramgen(param,res,true);
title('GA Parameter')
[m p t] = ppmetrics(par)

m =

    0.6916


p =

    0.8674


t =

    0.8684
```

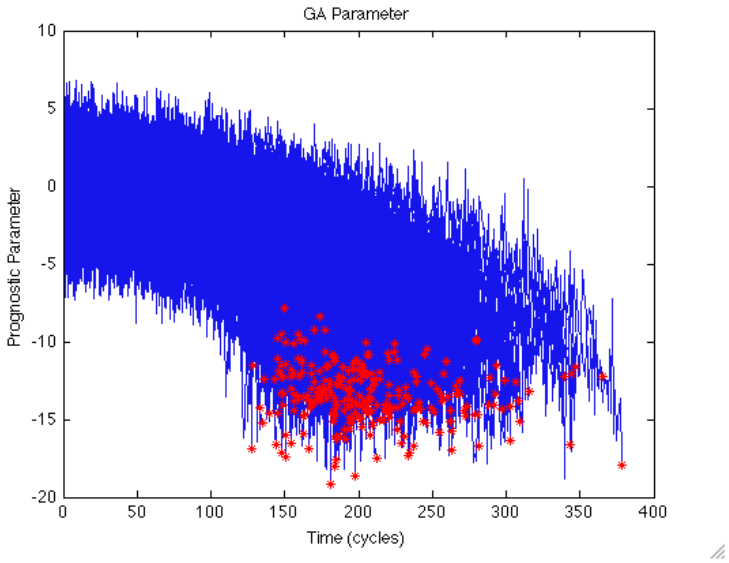


Figure 10.3-7: Training Degradation Paths

```
% A prognostic model is developed for the GA optimized parameter          %
typeIII = initprog('gpm',par,'bayesian',true)

typeIII =

              type: 'GPM'
          bayesian: 1
               fit: {[@(x)x.^2]  [@(x)x]  [@(x)1]}
        ytransform: @(y)y
         threshold: [-10.9743 0.5743]
     thresholdtype: 'pdf'
         threshcon: 0.9500
              npop: 1000
          noisevar: 2.0934
     bayesianprior: [2x3 double]
     allownegative: 0

% Monitoring system residuals for the test runs are obtained from the     %
% previously developed models                                             %
tst_g1 = cell(size(tst));
for i = 1:length(tst)
    tst_g1{i} = tst{i}(:,Groups{1});
end
res_tst = residgen(model,tst_g1);


%%

% Finally, the models are used to estimate the RUL of each test run with  %
% each model                                                              %
```

```
tst_par = paramgen(param,res_tst);
TypeIIIOut = rungpm(typeIII,tst_par);
RUL_typeIII = TypeIIIOut.RUL;

%%

MAPE_typeIII = nanmean(abs(RUL - RUL_typeIII)./RUL*100);

figure; hold on
plot(RUL,RUL_typeIII,'bo');
plot(0:200,0:200,'r--');
hold off
box
xlabel('Actual RUL (cycles)')
ylabel('Estimated RUL (cycles)')
title(['Type III RUL Estimation Error : MAPE = ' num2str(MAPE_typeIII)])
```

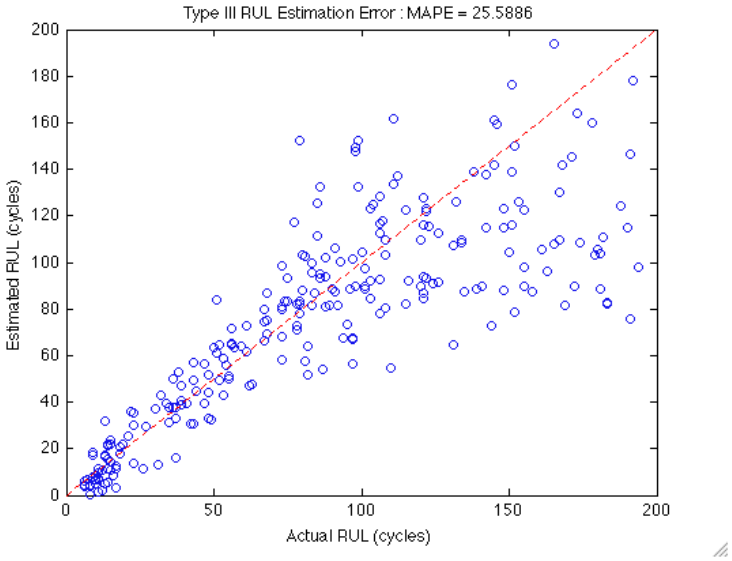


Figure 10.3-8: Type III RUL Estimation Error

```
%% Type III GPM with Bayes

% It is possible to combine a Type III GPM with a Type I or Type II prior
% model. This reduces the uncertainty of the RUL estimates by decreasing
% uncertainty on the path parameters.

TypeIIIBOut = runGPM(typeIII,tst_par,typeI);
RUL_typeIII_B = TypeIIIBOut.RUL;
```

```
MAPE_typeIII_B = nanmean(abs(RUL - RUL_typeIII_B)./RUL*100);

figure; hold on
plot(RUL,RUL_typeIII_B,'bo');
plot(0:200,0:200,'r--');
hold off
box
xlabel('Actual RUL (cycles)')
ylabel('Estimated RUL (cycles)')
title(['Type III w/ Bayes RUL Estimation Error : MAPE = '
num2str(MAPE_typeIII_B)])
```

10.4 Summary:

To summarize this section, several data sets were employed to test and validate the diagnostic and prognostic algorithms. The first data set consisted of simulated data that was used to test and validate the Bayesian transitioning algorithms. These transitions are used in prognostic model development, where the transitions are from Type I/II to Type III modeling. The reason for using these transitions is that with the Bayesian priors used along with the actual data there can be a large reduction in the error and uncertainty of the RUL estimates of the model. It was found that using Bayesian updating with prognostic parameters generated using OLS had the greatest effect in reducing the overall error in modeling. Next, the pump impeller degradation data was used to test and validate the prognostic model algorithms. The examples provided the needed information and code to generate each of the three prognostic model types. All models were validated by leaving one of the cases out, developing a model on the remaining cases and test on the left out case. The results show that with the inclusion of known operating conditions or degradation parameters of the system can reduce the overall error in RUL estimates than just by using a Type I model. Last, the PHM challenge data was used to further validate these prognostic algorithms. The modeling results for this data set are well documented in the PEP toolbox users' manual.

11. Concluding Remarks and Summary

This research covered a wide array of topics and experiments concerning the advancement of lifecycle prognostic modeling. Investigations ranged from the sources of uncertainty in modeling and how to best deal with them; when and how new data would be expected to become available at the different stages of a system's lifetime. As part of this, tools for utilizing this information in a concise manner were developed to aid future researches as well as industry and production managers in creating and maintaining the highest levels of quality, safety, and reliability within their facilities.

Where traditional prognostic analysis relied on separate styles of models, each based on the primary level of information expected to be available about a system throughout its lifetime, the algorithms and techniques presented in this paper provide clear methods for moving between different models as further information becomes available during the lifetime of a system. This allows for sensible expectations and decisions about a system to be made at any stage of that system's lifetime: prior to deployment, initial operations and startup, regular running operations, and especially near the expected end of component lifetime. Prior to the initial deployment, no specific information or expectations from a system are likely to be available. However, there must still be informed decisions made as to the expected operations of this system. During this phase, models must rely exclusively on information about prior populations of the system of predictive physics of failure probability models. Once information about the expected system environmental and duty stresses becomes available, this can be used to augment any population or physics based model predictions, updating them from general populous to reflect the most pertinent subset, effectively shifting any estimations on lifetime while also lowering uncertainty. During most normal operations and prior to any incipient fault or degradation detection, these models may provide the most reliable prognostic predictions. This is also the time when online monitoring of systems and components becomes usefull for the detection on these incipient faults. After a fault or anomaly has been detected, the final phase of prognostic modeling can be used to extrapolate or otherwise predict the progression of this failure to some point where the system will no longer meet design or safety criteria. Seamlessly incorporating all the information from previous predictions as well as this most up to date online information gives the most complete picture or a system's reliability so that logical, informed, and experienced decisions can be made.

The tools and algorithms developed to implement these procedures and models were put into a comprehensive MATLAB toolbox known as the Process and Equipment Monitoring and

Prognostics (PEMP) suite. Spanning all the phases on modeling, detection, and prediction during a system's lifetime, this suite of tools includes both inline algorithm coding tools and an interactive Graphical User Interface (GUI) for data analysis and prognostic model creation. These tools were tested on and validated with both simulated data and data collected from a wide array of physical accelerated aging test bed facilities located both at the University of Tennessee and across the US.

Three different test bed setups were constructed at the University of Tennessee, and utilized during the testing phases of this project. This first was an electric motor accelerated aging rig designed to thermally stress 5 horsepower motors. Over ten motors were cycled and aged to failure during the course of this project. Although most failures did not precipitate from the desired internal electrical failures, a testing procedure to help better in sure these types of failures in future experiments was developed and successfully tested.

Additionally, the university developed both a heat exchanger fouling experiment, and a pump impeller aging test facility. Data collected from both of these setups was instrumental in verifying and validating the tools and algorithms created over the course of this project. Examples and case studies of this data both with traditional analysis methods and those contained within the PEMP suite are presented in both the body and appendices of this document.

The validation of the PEMP suite of tools merited the investigation of existing forms of model performance metrics. This ultimately culminated in the development of a more flexible and intuitive set of model performance metrics that can quickly be used to compare and quantify the performance of any prognostic model. These new metrics have been publicly published and are available for general justification of model acceptance comparative analysis between models.

As the science of prognostics continues to develop and gain acceptance in more and more industries, proper handeling and presentation of data becomes ever more crucial. Having tools and people able to quickly and reliably analyze data is paramount for any complete reliability regeme. This project has developed tools and helped to train over eight students in the field of reliability and prognostics.

12. Future Work:

After the completion of this project, clear indications of the next stages of research and development present themselves. Firstly, supplementary development and validation of the PEMP suite toolbox on additional data sets from both controlled experimental test beds and eventually true industry data would be invaluable to the furthering of this work. While collection and testing of the algorithms on small to medium scale test bed applications is incredibly useful for verification of fault detection and isolation algorithms, it will eventually require interfacing with actual plant systems and data in order create a fully evaluated, deployable system ready for online plant operations.

Other, more specific paths of future work include the further algorithm and toolset development via such additions as the implementation of models with separate forms of reliability indication within the PEMP suite. Currently, most models are set to indicate a quantitative value of Remaining Useful Life (RUL) with some measure of uncertainty associated with it in terms of operational cycles (such as minutes, hours, days, etc.). In certain cases, it might be more useful and informative to provide the optional indications of either direct percent of life consumed at a given time, or the probability of failure at some query time. These additional measures of reliability could help to better inform operators when making operational decisions about a system.

Additional options for incorporating event frequency and/or qualitative forms of expert provided information beyond what is directly sensed on the system would also provide a unique opportunity to advance the algorithms and utilize the fullest amount of information available. Often, expert operators fill out operations and maintenance logs or qualitative performance evaluations that could contain pertinent information related to the stresses and overall health of the system. If future incorporated toolsets could extract important patterns and information from these or similar sources and quantify them into useful indicators, this information could be useful in augmenting existing predictive models and evaluation tools to give the most robust and accurate picture of a system's overall health. Automated methods for the extraction and evaluation of these patterns, such as a bag of words technique, should bean integral part of this toolset.

Beyond the one-dimensional sets of signals typically associated with systems' sensor logs, ways of evaluating multidimensional sensor or signal inputs, such as thermal camera imaging, joint

time-frequency mapping, or general video imaging of component operations should be incorporated into a complete prognostic and reliability modeling suite. These interrogative and evaluative methods produce a series of multidimensional information maps that relate specifically to a single (or multiple) sensed system or component. These require specialized techniques for pertinent feature identification, extraction, and modeling. Combining this information into models built around multivariate collections of one-dimensional variables may ultimately require the collapse of these multidimensional inputs into a similar series of one dimensional indicators. Future work should also center on developing an automated, robust, and reliable algorithm for accomplishing this that maximizes the pertinent information and also minimizes transference of unnecessary information or noise. Areas of initial investigation for this topic could include image processing and compression techniques such as independent component analysis or in the case of time frequency maps, peak identification and tracking.

When considering the motor degradation experiment, there are some modifications that could be made to the testing plan to obtain a larger set of useful degradation information. The first is to use an overloading method to degrade the motors; using thermal and moisture cycles in the first and second years of the experiment did not yield the types of failures as expected. When using the thermal degradation method it was seen that the main failure mode was in the wire connections to the stator, this caused the motors to fail even though several more tests could have been run if this fault had not occurred. It also did not provide any prognostic information. Our goal was to see stator and rotor winding failures, but we were not able to force those failure modes. The overloading method heats the motor during operation by increasing the load output on the resistor bank. This forces the heat source to be inside the windings, where the degradation is desired. The data analysis on the two motors that utilized this degradation method showed that there were noticeable changes in the power output of the motor due to increased temperature and resistance from the overload conditions. This type of profile was not seen in the earlier test runs because the motors because the wire connections to the stator shorted out before this desired failure could occur. Future tests could have motors run at different overload conditions for extended periods of time to determine if internal wiring failure modes might manifest earlier in the data or extracted features so that refined RUL estimates could be obtained. We are proceeding with overload tests with funding from Dr. Jamie Coble's startup package. The impact of this for a real world application is that maintenance strategies can be modified so that extended downtime, repair costs or injury can be avoided.

There are also modifications that could be made to the pump impeller experiment that were not used during the project. During the experiment it was seen that there was a wide range in the failure times even though most impellers came from the same batch and all were thermally aged in the same manner. In the future, the thermal aging procedure could be modified to test impellers that are aged at a slightly lower temperature but for extended periods of time. Using this modified thermal aging plan might cause the failures to occur in roughly the same time span instead of the wide variance that was observed. Additionally, different impeller materials such as nitrile could be used to see if the same failure modes and features are seen as in neoprene. This would serve to create a database of degradation information for several impeller types which could be used in a variety of industries. Lastly, difeent types of pumps or difeent ageing mechamisms could be utilized.

Another useful future modification for an impeller aging test bed would be to add a blockage into the inlet line along with the blockage in the outlet line that was used during the project. The application would be to determine if the additional blockage or combination of line blockages reduces or increases the expected time to failure. Also, the correlations between the different blockages could be examined to optimize a testing plan. Finally, different fluids at varying temperatures and with additives could be examined to determine their effect on failure time. As in the motor degradation experiment, accelerated aging using a variety of conditions helps to develop a database of degradation data with a wider range of applicability for prognostic modeling and maintenance planning.

For the heat exchanger, possibly the most important future research needs to involve the broadening of the training and testing data over different flow rates, including hybrid cycles that operate at multiple flow rates over time. To improve lifecycle prognostic model results, additional data at the current flow rates should also be taken to improve model robustness to different operating conditions. One area of future work that needs focusing is the redesign of of the heat exchanger construction. Research should be done on an updated test bed setup that reduces uncontrolled variables such as cold water input temperature, hot leg pressure, and clay mixture concentration.

Another area of future research that should be investigated is the construction of a natural flow loop accelerated degradation test bed. The impact would be a better inderstancing of natural circulation loops such as those in NuScale and inherently safe emergency cooling systems. Heat exchanger operation and data relationships are significantly different for forced and natural

circulation flow. The degradation paths seen in a natural circulation loop will likely differ substantially from those that were measured in the current heat exchanger test bed. Because there is no artificial circulation, the fouling of the heat exchanger tubes could potentially greatly reduce flow as the particulates accumulate within the natural circulation heat exchanger. These unique degradation paths would require a new lifecycle prognostic model. Lastly, future efforts should be focused on the aggregate bootstrapping of residual/signal values from competing models to serve as a variance reduction technique. To achieve this, several residuals of the same type from different models (such as the hot leg inlet temperature across 4 models) would be combined in some way such as linear merging or averaging. This modified residual could then be used with other residuals to build the prognostic parameter. This helps to reduce variance across models.

The results of the General Path Model using Type I and Type II priors has already shown great promise in reducing the uncertainty of RUL estimates resulting from sensor noise and degradation model. The Bayesian framework developed and implemented covers most instances of Type I to Type II and Type III transitions. However future research can explore additional methods for combining multiple types of prognostics with and without the use of Bayesian statistics. In developing more Bayesian methods, additional exploration could be done on controlling the weights of the prior to more suitably stabilize RUL estimates for the GPM. For example, a more careful consideration of the weight of a Type I prior can be made in relation to the amount of data that exists for the GPM. For datasets that involve very long life times, and thus large amounts of GPM data, the Type I prior can be quickly overpowered and result in not much change from using only the GPM. In such instances the Type I prior must be made stronger to express more confidence relative to the initial life of the system. The other extreme can also be considered. If the Type I prior is weighted too heavily, the advantages of the GPM can be lost. As such, expert knowledge of the system and the mathematical subtleties of Bayesian statistics are required for a useful implementation of the Bayes transition methods. However, this ambiguity must be addressed in the context of empirical modelling. In other words, there should be better methods of developing such finely tuned prior weights that depend more on the data than on the user. Such methods can involve simple solutions such as a single equation, or set of equations, that can be applied to any dataset that would balance out the prior with the posterior. Or a set of equations that requires one or two user inputs, such as a basic confidence comparison of the Type I to the Type III, i.e. if there exists a large amount of Type I information, but relatively little is known about the Type III GPM of a system, then there should be much more confidence in the Type I prior. Conversely if a system's RUL can be accurately obtained through the GPM, then

there is little need to express confidence in the prior, other than in the beginning of life. Other advanced methods can involve optimization methods on known failures that will select a prior weight that minimizes the RUL error based on the beginning of life, end of life, and the number of data points that exist in one lifecycle.

13. References

Brear JM and PF Aplin. 1994. Life Management of Power Plants, 1994., International Conference on, pp. 108-113. 12-14 Dec 1994.

Burchill, R.F. (1964). Resonant Structure Techniques for Bearing Fault Analysis. Proc. 18th Meeting MFPG. Gaithersburg, MD: National Bureau of Standards.

Coble, Jamie Baalis, "Merging Data Sources to Predict Remaining Useful Life – An Automated Method to Identify Prognostic Parameters. " PhD diss., University of Tennessee, 2010.

Courrech, J., & Eshleman, R. L. (2002). In C.M. Harris & A.G. Piersol (Eds.), Condition Monitoring Of Machinery (16.9). New York: McGraw-Hill

Hosford WF. 2005. Mechanical behavior of materials. Cambridge University Press. ISBN 0521846706.

Li J and A Dasgupta. 1993. "Failure-mechanism models for creep and creep rupture." Reliability, IEEE Transactions on 42(3):339-353.

Nam A, M Sharp, JW Hines and BR Upadhyaya. 2012. "BAYESIAN METHODS FOR SUCCESSIVE TRANSITIONING BETWEEN PROGNOSTIC TYPES: LIFECYCLE PROGNOSTICS " In 8th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC & HMIT 2012). July 22-26, 2012, San Diego, CA. American Nuclear Society.

Naumenko KD and H Altenbach. 2007. Modeling of creep for structural analysis. Springer. ISBN 3540708391

R. Haupt and S. Haupt, "Practical Genetic Algorithms", Wiley & Sons, 2nd Ed., 2004.

R. Li, P. Sopon and D. He, 'Fault features extraction for bearing prognostics", Springer Online Text, 2012.

Raj B, CK Mukhopadhyay and T Jayakumar. 2006. "Frontiers in NDE Research Nearing Maturity for Exploitation to Ensure Structural Integrity of Pressure Retaining Components." International Journal of Pressure Vessels and Piping 83(5):322-335.

S.A. McInerny and Y. Dai, "Basic Vibration Signal Processing for Bearing Fault Detection" , IEEE Transactions on Education, Vol. 46, No. 1, 2003.

Shao, H., Jin, W., Qian, S. (2003). Order Tracking by Discrete Gabor Expansion. IEEE Transactions on Instrumentation and Measurement. 52. 3. IEEE.

Sharp, Michael E., "Prognostic Approaches Using Transient Monitoring Methods", A Doctorial Dissertation, The University of Tennessee, Knoxville TN. August 2012

Sharp, Michael E., "Advanced Standardized Metrics for the Intuitive Evaluation of Prognostic Model Performance Across General Applications." International Journal of Prognostics and Health Management, 2013.

Sposito G, C Ward, P Cawley, PB Nagy and C Scruby. 2010. "A review of non-destructive techniques for the detection of creep damage in power plant steels." NDT & E International 43(7):555-567.

Strong, Eric, "Development of a Methodology for Incorporating Fault Codes in Prognostic Analysis," *Dissertation University of Tennessee*. 2014.

Xuedong Huang, "Theoretical and experimental study of Degradation monitoring of steam generators and heat exchangers", The University of Tennessee, Knoxville, 2003

14. Appendix 1: Additional Heat Exchanger Figures

14.1 Plot of Features

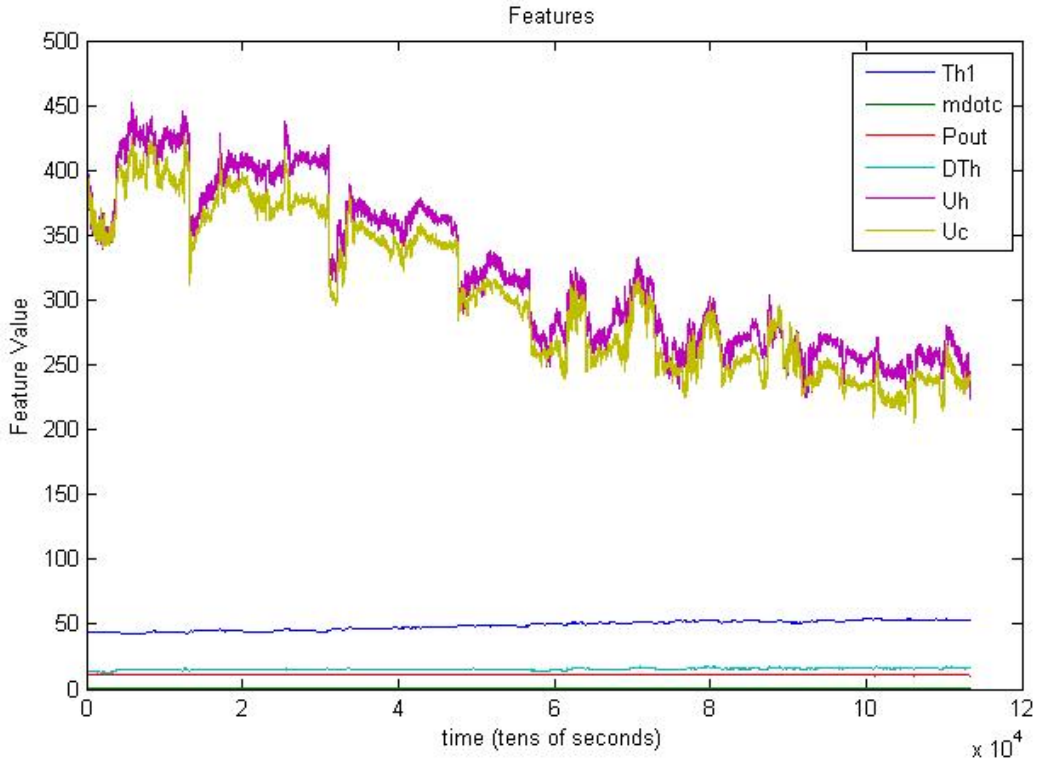


Figure 14.1-1 Example plot of major features for a single cycle after cleaning

14.2 Summary of Signal Indices

Table 14-1 Table of signal indices and their corresponding signal titles

Signal Index	Signal/Feature
1	Hot Leg Inlet Temperature
2	Hot Leg Outlet Temperature
3	Cold Leg Inlet Temperature
4	Cold Leg Outlet Temperature
5	Hot Leg Flow Rate
6	Cold Leg Flow Rate

7	Hot Leg Inlet Pressure
8	Hot Leg Outlet Pressure
9	Delta Hot Leg Temperature
10	Delta Cold Leg Temperature
11	Hot Leg Heat Rate
12	Cold Leg Heat Rate
13	Log Mean Temperature Difference
14	Hot Leg Overall Heat Transfer Coefficient
15	Cold Leg Overall Heat Transfer Coefficient

14.3 Code for LabView Extraction

```
function [FinMat] = LVextract(directory,depvar)
% function to strip data from LABVIEW excel files and combine into a final
% matrix, and displays the variable names by column.
%---To pass
%   directory use-
%                   directory = uigetdir;
%
% which will allow user to GUI to the directory containing all excel files.
%
% ---depvar is the number of dependant variables not to exceed 25.
%
%                           By:  Zach Welz
%                           University of Tennessee Knoxville

disp('Warning: Variable names are taken from last file in directory.')
 a = 1;
```

```
num = 1;
old_directory = pwd;
files = dir(fullfile(directory, '*.xlsx'));
fileIndex = find(~[files.isdir]);
for i = 1:length(fileIndex)
ins = 1;
fileName = files(fileIndex(i)).name; % creates file name
cd(directory); % changes directory to where excel copies are (specified by the user)
%% Get length of sheet 2
sheet = 1; % length is listed on sheet 1
xlRange = 'D9'; % in cell D9
sheetlength1 = xlsread(fileName, sheet, xlRange); % gets cell D9 information
sheetlength2 = sheetlength1 + 1; % accounts for the header line
%% Get data from sheet 2
sheet = 2; % data in contained on sheet 2
intro = 'B2:I'; % beginning of cell range
celldef = [intro,num2str(sheetlength2)]; % completes cell range string
xlRange = celldef; % defines final cell range
sub = xlsread(fileName, sheet, xlRange); %strips data from file
%% Get headers from sheet 2
var1 = 'A':'Z';
spread = ['B1:',var1(depvar+1),'1'];
[~, ~, firstrow] = xlsread(fileName, sheet, spread);
%% store data in full matrix
b = a + sheetlength1 - 1;
if num == 1
  for j = a:sheetlength1
    for k = 1:depvar
      FinMat(j,k) = sub(j,k);
    end
  end
else
  for j = a:b
    for k = 1:depvar
```

```
        FinMat(j,k) = sub(ins,k);
      end
     ins = ins + 1;
   end
end
num = 2;
olda = a;
a = sheetlength1 + olda;
end

%% Clean Data
filtbinwidth = inputdlg('Input the desired bin width for cleaning.');
filtbinwidth = str2double(cell2mat(filtbinwidth));
FinMat = medfilt1(FinMat,filtbinwidth);
% FinMat = cleandata(FinMat); % use if prefer cleandata cleaning method
%% Show Resulting Variabels
va = 1:depvar;
for i = 1:depvar
   list{i} = [num2str(va(i)),'      ',firstrow{i}];
end
msgbox(list)
 cd(old_directory);

return
```

15. Appendix 2: PEMP Suite Users guide

Welcome to the Process and Equipment Monitoring and Prognostics GUI. What follows is an example walk through showing some of the functionality of this software.

15.1 STEP 1 – Import and Separate Data

All data must be preloaded into the MATLAB workspace with the general form that row index indicates observation or case number and column index indicates signal number oradditional data segemts.

In this example the data is divided into 3 variables:

UnFaulted – This cell contains 50 example cases of falut free runs with this system.This will be used to create the monitoring model.

FailureData – This is a cell that contains 50 separate cases of the system recorded from beginning of life to failure. This will be used to develop the prognostic model.

LifeTimeData – This is a matrix with the total length of 50 additional systems were in operation before failure is time units similar to those of the previous two variables. This will be used to augment and enhance the prognostic model.

The final variable in the workspace is the QueryData. This is data in a similar format to the FailureData variable and will be what the models will be run on after fully constructed.

Each variable is shown in Figure 15-1. Also in this figure is shown the internal structure of the one of the cells. In this case the system has 7 data signals represented by the 7 columns in each cell. These signals correspond to 3 electrical signals, 2 flow sensors, and 2 vibration signals on a simulated recirculation pump flow loop.

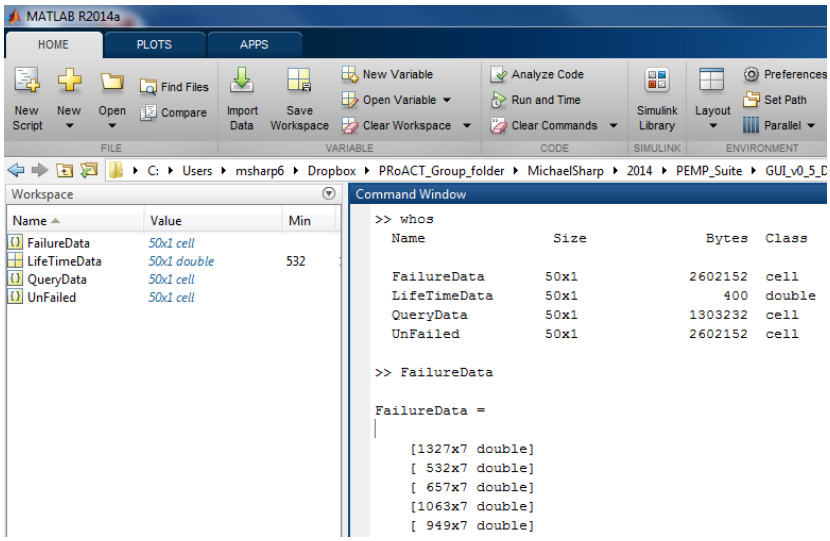


Figure 15-1 : Beginning MATLAB Workspace for GUI

Once the data is properly loaded into the workspace then the Start_Here script can be run regardless of the working folder or directory. If a project structure has previously been created, then the only thing necessary to load to the work space in addition to the query data is the project variable. Each individual project created is unique to the data from the system used to create the project models.

15.2 Step 2 – Create a New Project

In Figure 15- is the initial panel view of the PEMP Suite GU interface which can be activated by running the Start_Here script after adding it and all necessary functions to the MATLAB working path.

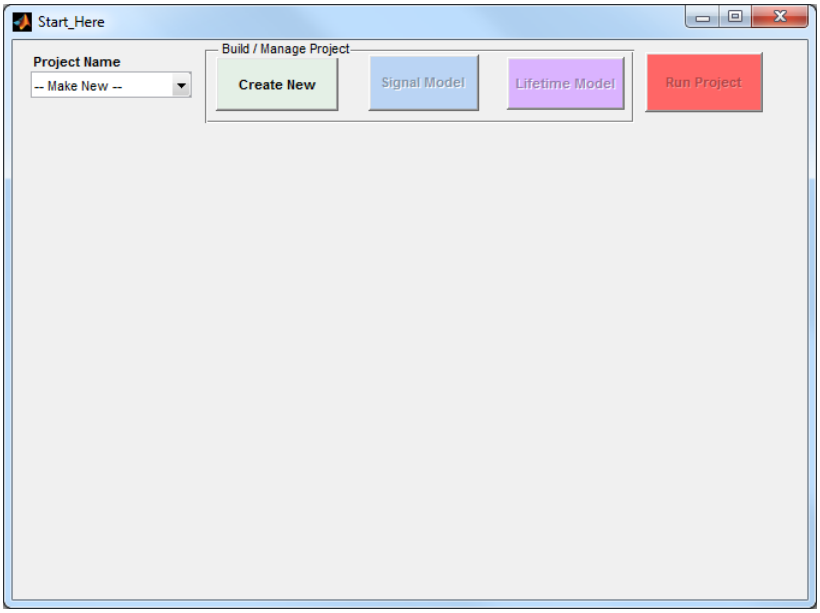


Figure 15-2: PEMP Suite GUI Initialization

Once this has been accessed, the next step is to either load the project of interest from the workspace by selecting it from the **Project Name** dropdown menu, or to click the **Create New** button. Note that on different operating systems the button colors my not be the same as shown in the figure above.

After clicking on the **Create New** button, a new dialog panel will appear as shown in Figure 15.2-. This panel allows for the naming of a project as well as the specification and input of the data needed to create both the signal prediction and the prognostic lifetime prediction models that will be used to make up the PEMP Suite GUI project. Under the **Unfaulted System Signals** menu, select the workspace variable that contains the fault free data that encompasses the expected operating ranges of the system data.

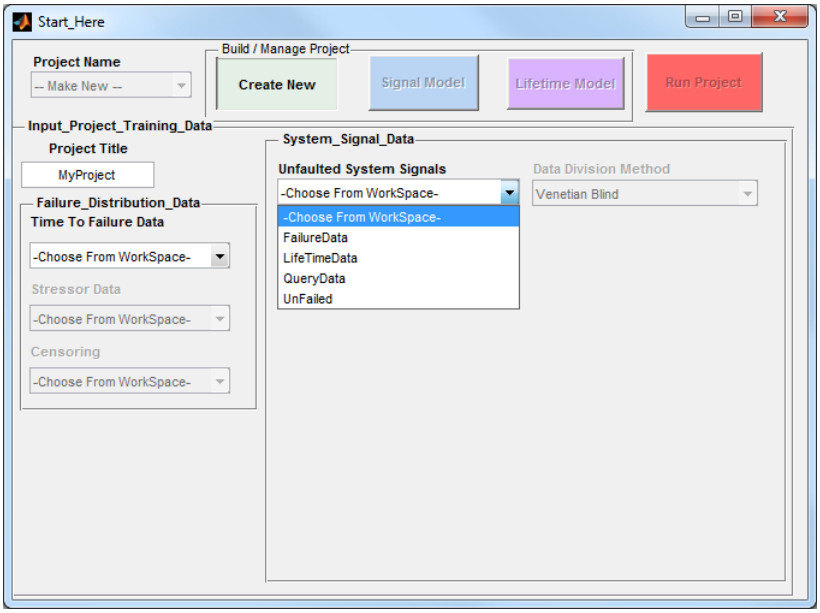


Figure 15.2-3: Create New Project Panel - Monitoring Model Data Input

Next, as shown in Figure 15-, select the method for dividing the data within this variable into 3 further sub-groups that will be used to build, test, and validate the model respectively.

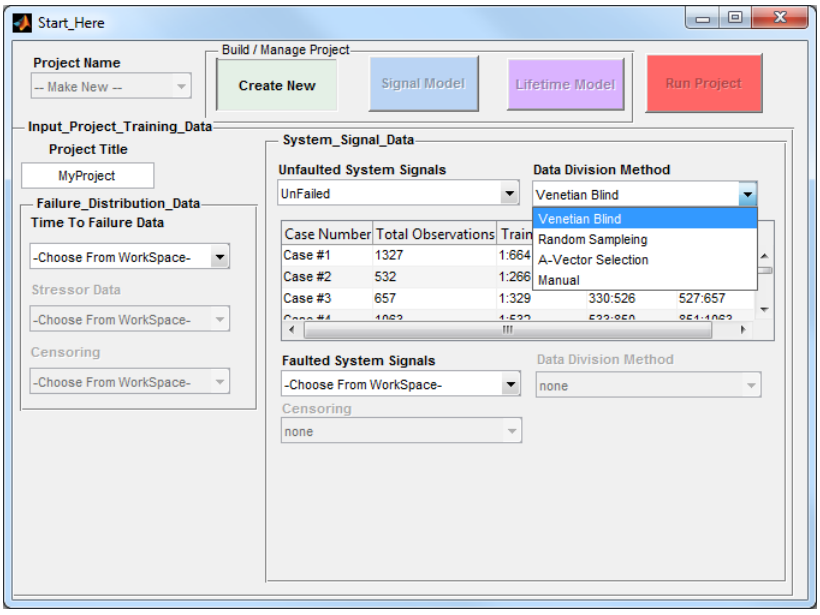


Figure 15-4: Create New Project Panel - Monitoring Model Data Division Method

Similarly, shown in Figure 15-, menus will appear that allow you to input and divide the workspace variable that contains observed runs to failure of the system signal sets which will be

used to create the prognostic model. A vector containing a distribution of known failed lifetimes can also be input in this panel to augment the lifetime predictive model.

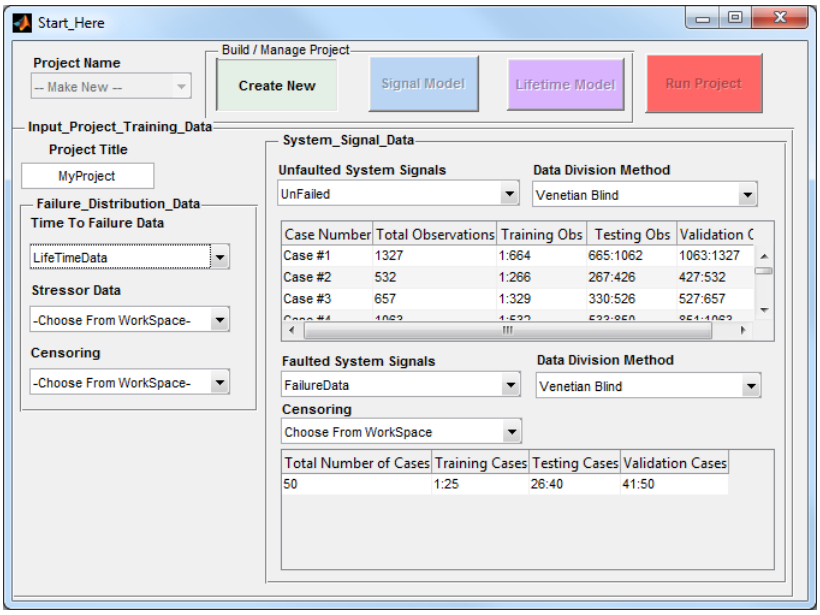


Figure 15-5: Create Project Panel - Prognostic Model Data Inputs

15.3 Step 3 – Build Signal Monitoring Model

After inputting all the appropriate data, the next step is to de-select the **Create New** button and select the **Signal Model** button. This will bring up the signal model construction panel as shown in Figure 15.3-.

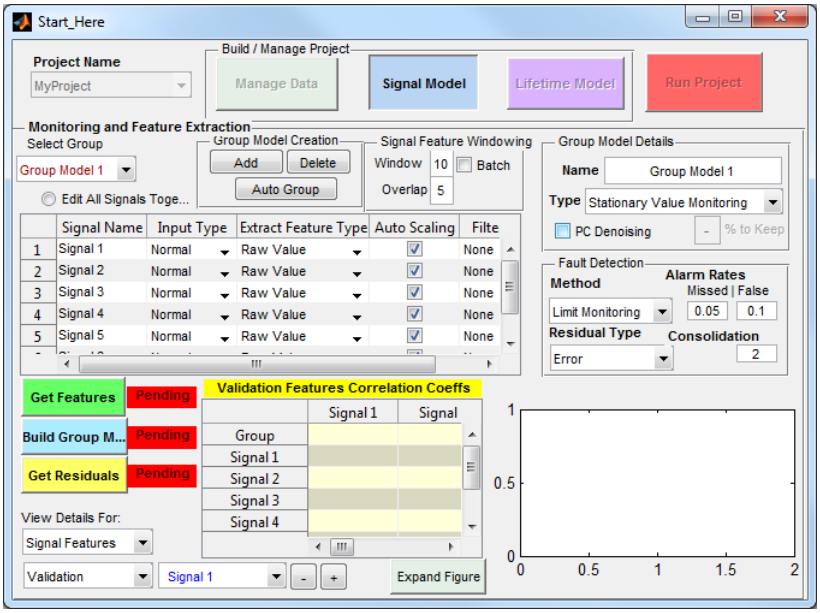


Figure 15.3-6: Signal Monitoring Model Creation Panel

This panel controls all the options for feature select, model specification, residual monitoring and fault detection for the systems signals. For this example, not all the features are explored, but the primary outline of model construction is explored.

This first step in model construction is to group highly correlated or related features extracted from various signals into sub models. This can be done manually by selecting the features of interest in the dropdown menus as shown in Figure 15-, or by pressing the **Auto Group** button. Each feature is processed in windows governed by the **Window** and **Overlap** input boxes above the interactive signal feature extraction matrix.

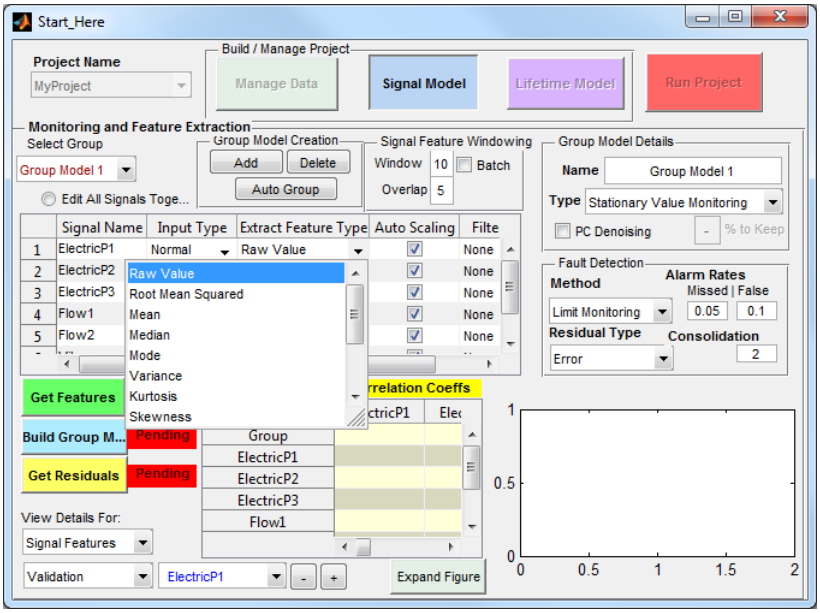


Figure 15-7: Signal Monitoring Model Creation Panel - Signal Feature Selection

This interactive signal feature extraction matrix allows not only for feature extraction specification but also allows for scaling, filtering, and naming of individual signals. By selecting the **Edit All Signals Together** radio button, any change in the matrix applied to one signal is similarly applied to each other signal. The **Input Type** sub box in this matrix allows for defining a signal as "Normal", "Time", "Stressor", or "Not In Group". By default, all signals are considered "Normal", indicating that they are both an input and a requested output of the model. The first (highest) Time signal indicated in the matrix will be used to define the online timing of the signals within the matrix. By default the observations are assumed to have uniform unit spacing. A denotation for a signal as "Stressor" indicates that this signal is an input to the model, but is not to be considered as an output. Finally, signals listed as "Not In Group" will not be considered as inputs or outputs of the current sub model and be ignored. Changing between sub models can be done with the **Select Group** dropdown menu.

Once all the features and signal processing requirements have been properly specified, clicking the **Get Features** button will process the model input signals and display them in the lower plot as shown in Figure 15-.

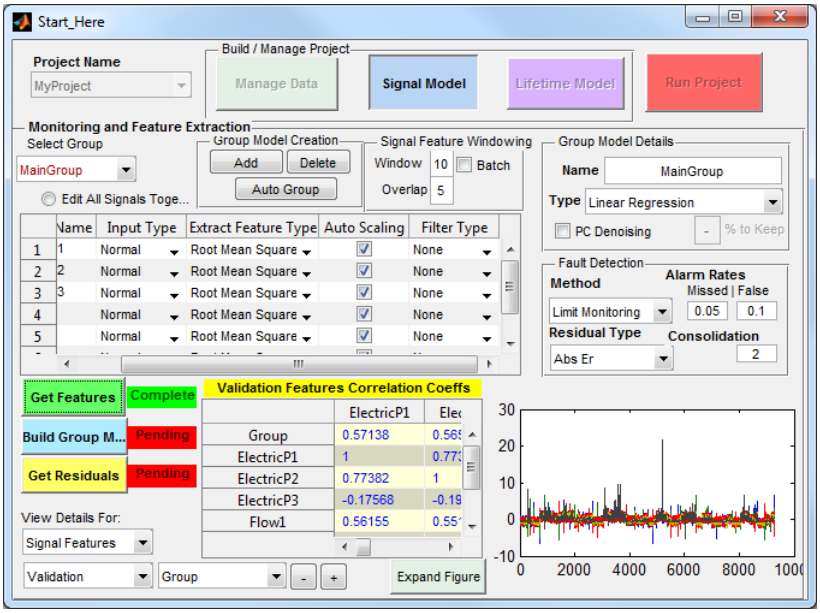


Figure 15-8: Signal Monitoring Model Creation Panel - Feature Extraction

The lower output chart also will indicate the correlation coeffecents between the system signal features.

After the features have been extracted, the type of model for the active sub model can be seleccted as indicated in Figure 15-.

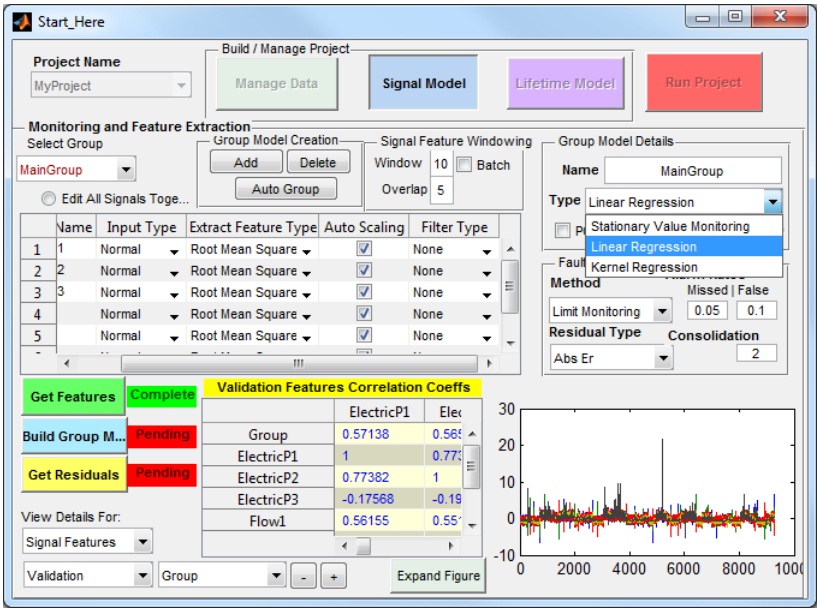


Figure 15-9: Signal Monitoring Model Creation Panel - Model Specification

These models may take some time to create depending on the model specification and the operating platform, but shortly after clicking the **Build Group Model** button a wait-bar should appear.

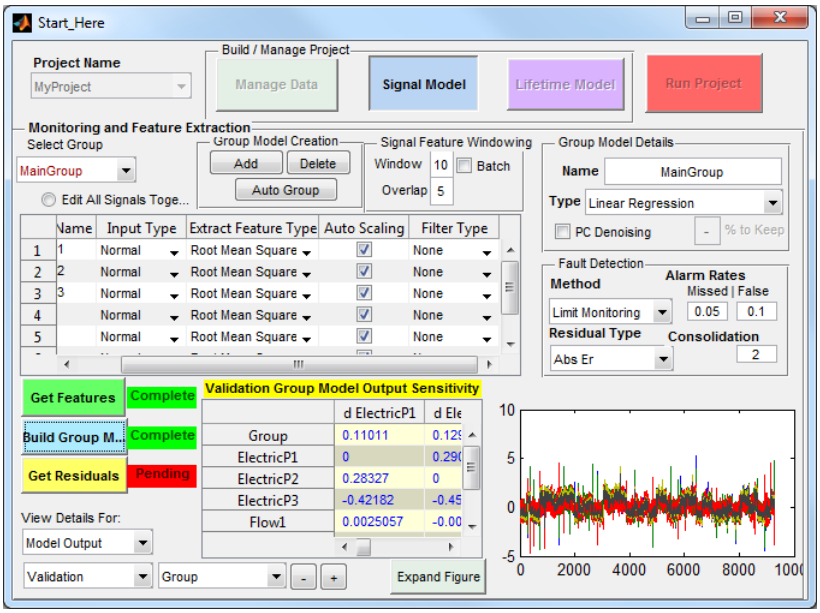


Figure 15-10: Signal Monitoring Model Creation Panel - Model Construction

A graph of the model output as well as a chart of signal sensitivity within the model will appear after the model construction is complete as shown in Figure 15-.

The final phase of the process signal monitoring model construction is to specify the type of residual monitoring and fault detection. These will be used to specify the degradation monitoring inputs for the lifetime prognostic model in the next step. Figure 15- shows an example of these residuals, which are the processed difference between the model output value of the signal features and the actual signal feature value.

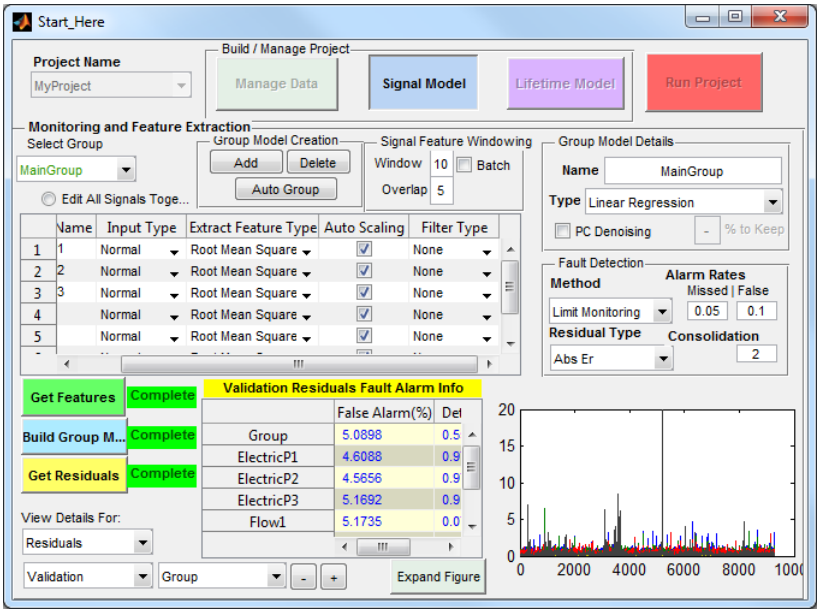


Figure 15-11: Signal Monitoring Model Creation Panel - Residual Construction

The fault alarms are also based on these calculated residuals and are shown in Figure 15-.

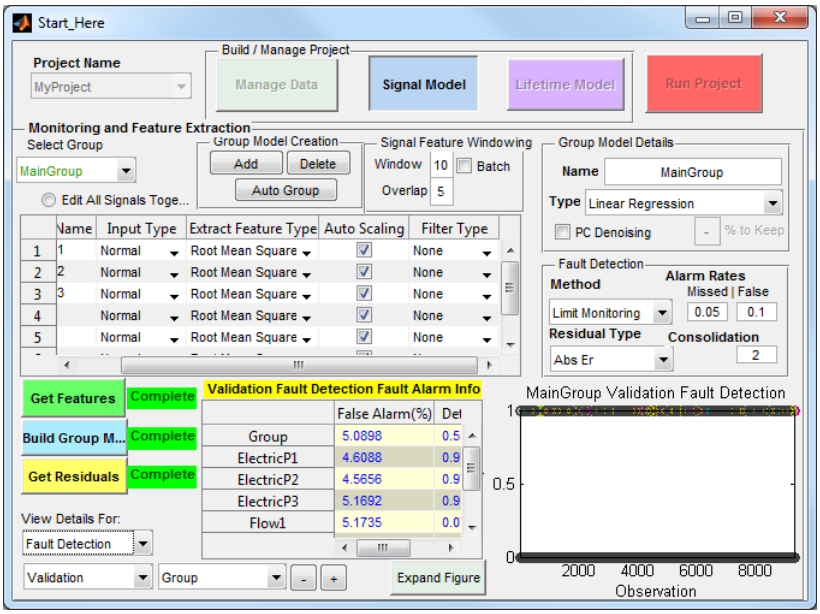


Figure 15-13: Signal Monitoring Model Creation Panel - Fault Alarms

Alarm statistics broken down by signal can be found in the displayed output box.

15.4 Step 4 – Prognostic Lifetime Prediction Model Creation

After creating and finalizing each of the sub models in the Signal Modeling Panel, deselect the **Signal Modeling** button and select the **Lifetime Model** button to activate the prognostic model construction panel seen in Figure 15-.

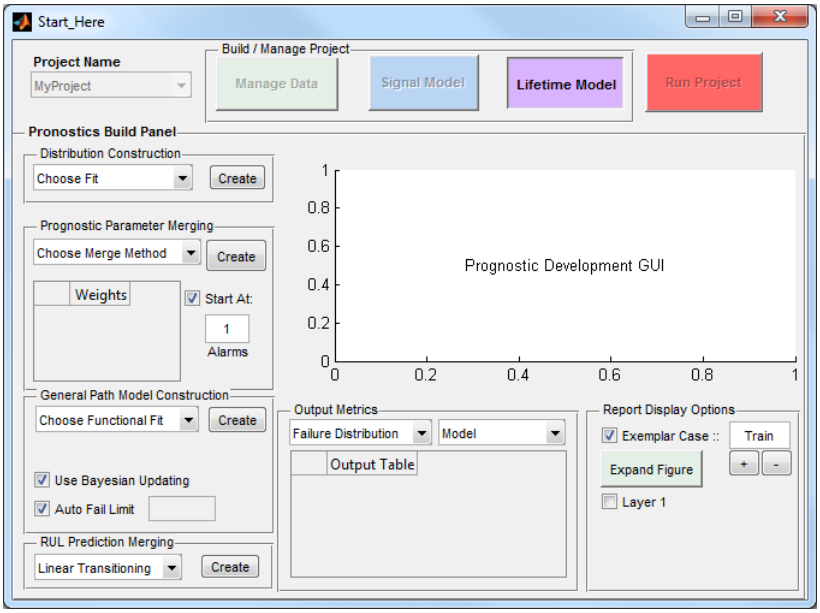


Figure 15-13: Prognostic Model Creation Panel

From this panel a failure time distribution model can be specified and created as shown in Figure 15-. This model will be used to create time to failure predictions prior to any initiating fault detection in the system as indicated by the Signal Model. It can also be used throughout the lifetime of the query to augment the general path model predictions.

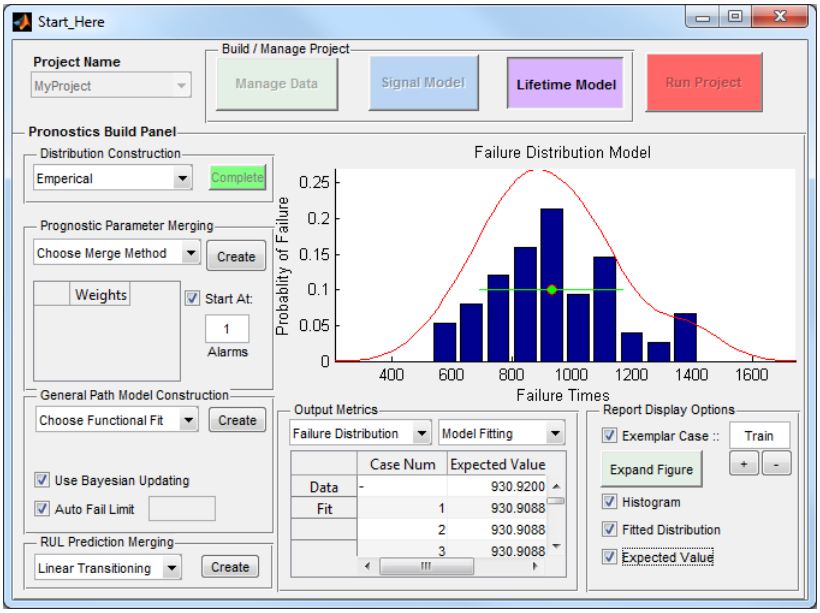


Figure 15-14: Prognostic Model Creation Panel - Failure Distribution Model Construction

Next a method for combining the system signal model residuals into a single indication of system degradation must be selected an implemented as in Figure 15-.

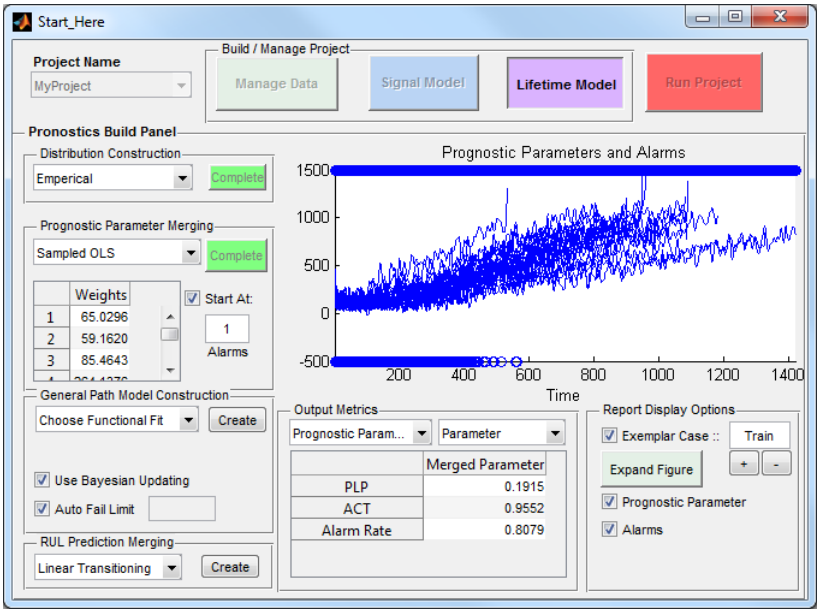


Figure 15-15: Prognostic Model Creation Panel - Finalized Prognostic Parameter

Some of the methods, particularly Genetic Algorithms and Gradient Descent, can by computationally intensive and therefore time consuming.

Once the system degradation parameter has been formed, the general path model form can be specified and implemented. Figure 15- shows the fitted pathways of this process overlaid with the corresponding degradation parameter.

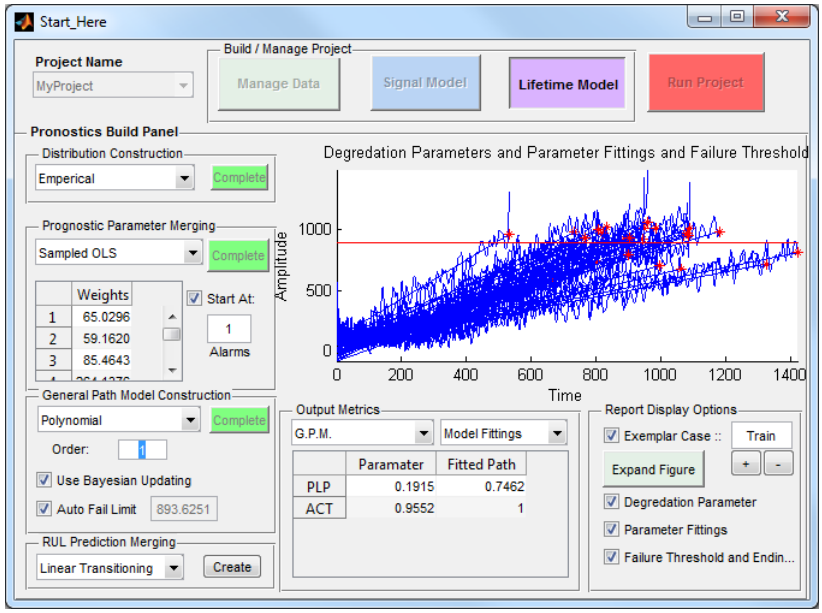


Figure 15-16: Prognostic Model Creation Panel - General Path Model Construction

The last step of project construction is to select how to merge the final output from both the general path model and the time to failure distribution model. Once that is done, as shown in Figure 15-, the performance of the model on the pre-selected validation cases should be reviewed to indicate performance.

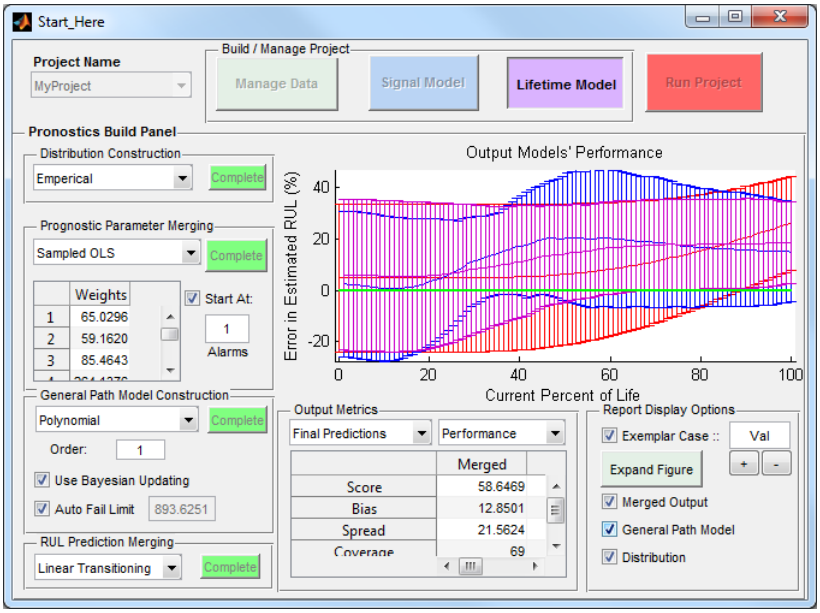


Figure 15-17: Prognostic Model Creation Panel - Merge Predictive Models Outputs

Individual cases of interest may also be reviewed for each stage in the prognostic model development process as indicated by Figure 15.4-1.

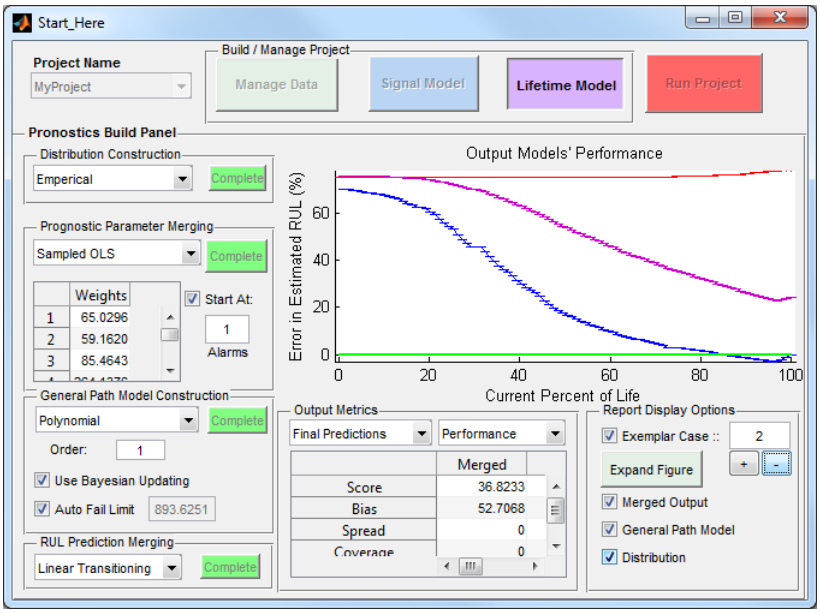


Figure 15.4-1: Prognostic Model Creation Panel - Examination of Single Case

15.5 Step 5 – Run Query Data

After completing construction of the PEMP project, the **Run Project** button becomes active and brings up the panel shown in Figure 15-.

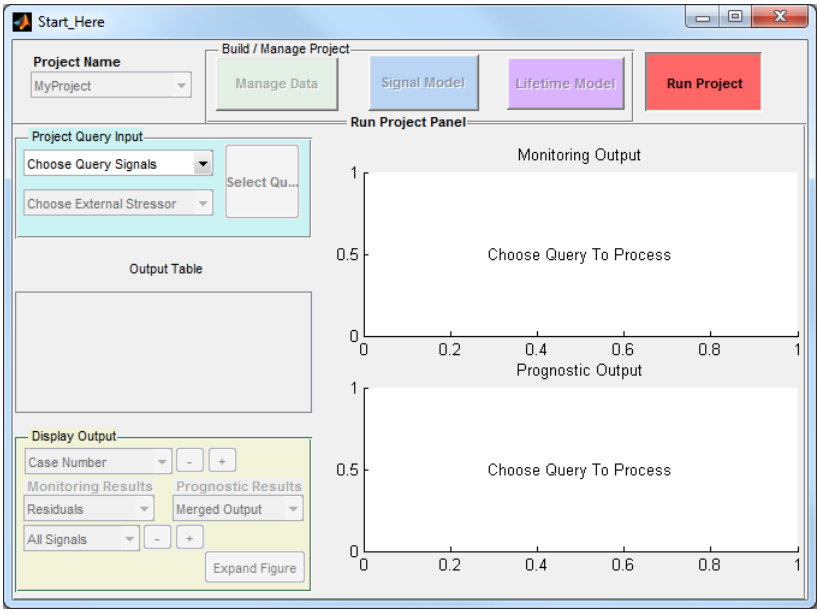


Figure 15-19 - Run Query Data Panel

After selecting an running the query data that was pre-loaded into the MATLAB workspace, the GUI presents an easily manages interactive way to navigate through the output of the various models within the project. Figure 15- shows an example of the model residuals and fault alarms as well the estimations for both the Lifetime Distribution model and the general path model (GPM) and their associated uncertainties.

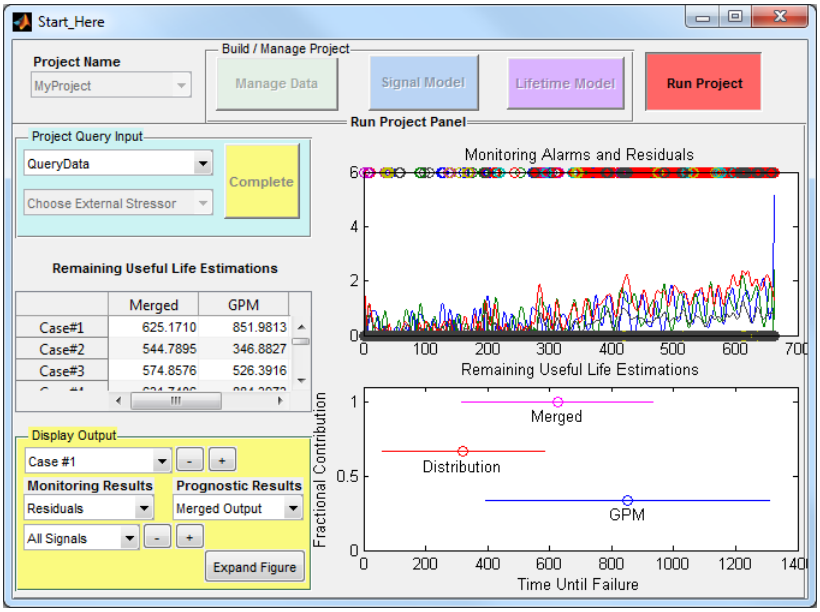


Figure 15-20: Run Query Data Panel - Monitoring Residuals and Prognostic Predictions

Further details of each of the models can also be gained as shown in Figure 15-, which contains both the process model directly predicted output and independently, the fitted path of the general path model.

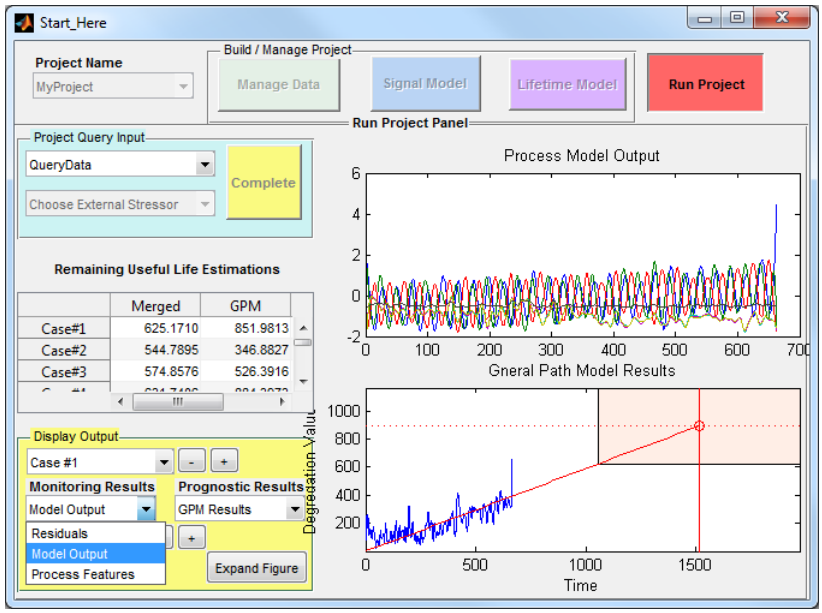


Figure 15-21: Run Query Data Panel - Signal Model Predictions and General Path Model Predictions

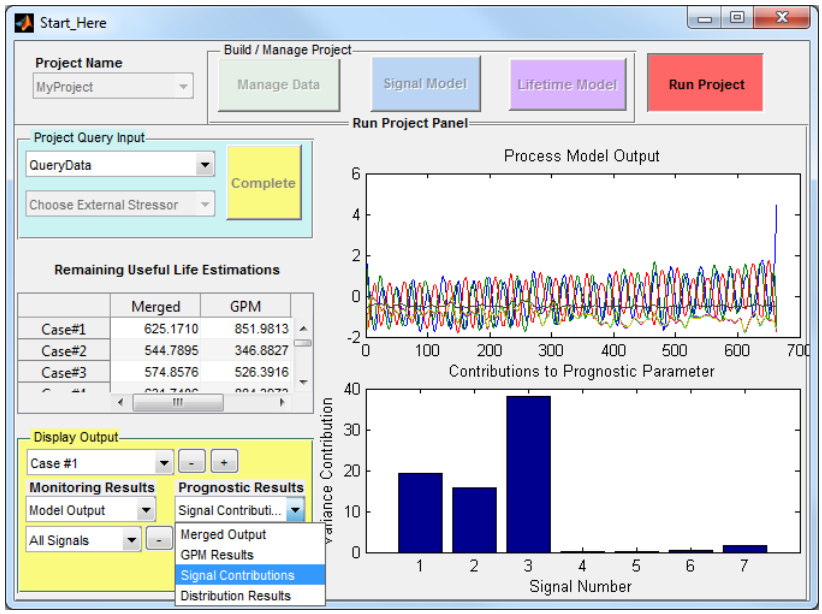


Figure 15-22: Run Query Data Panel - Signal Model Residual Contributions for Fault Identification

Another important feature of the GUI is that is can be used to indicat the realitive contribution each signals' residuals are providing to the overall degredation parameter used by the general path model. An example of this is provided in Figure 15-. This can be potentially helpful fault identification or classification.

Lastly, the PEMP Suite GUI saves a model output structure to the MATLAB workspace for easy use in other proograms or for further custom analysis. Figure 15- show an example of this.

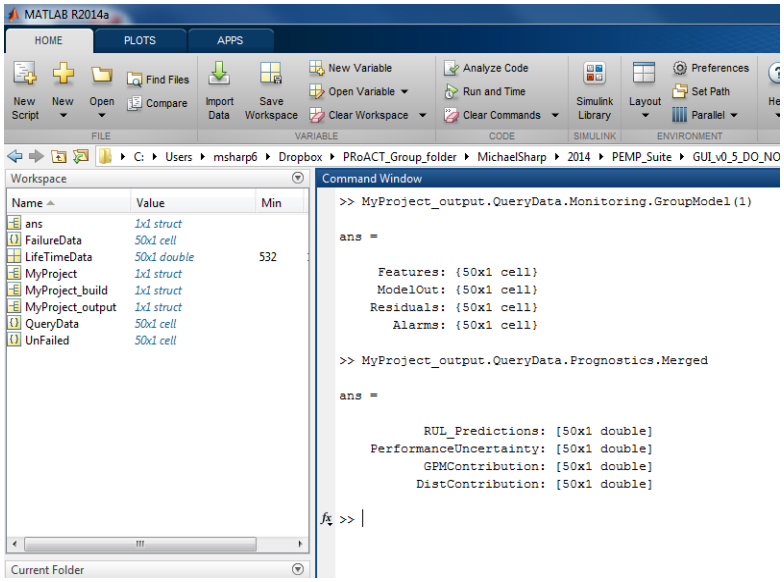


Figure 15-23: MATLAB Ending Workspace and Query Data Output Variable

The final output structure follows the naming scheme of the project name followed by a "_output" indicator.

16. Appendix 3: PEM and PEP Users Guide

16.1 Introduction

The Process Equipment Monitoring (PEM) and Process Equipment Prognostics (PEP) Toolboxes are MATLAB-based tools that facilitate condition monitoring and fast prototyping of empirical-based prognostic models developed at the University of Tennessee Nuclear Engineering Department. The goal of this User's Guide is to present the pertinent background information and basic syntactic information needed to employ the Toolboxes.

This guide begins by providing an introduction to the Toolboxes, including an overview of the toolbox architecture and requirements for employing the available functions. The relevant background of prognostics and the empirical techniques available in is summarized with appropriate references for the interested reader to find additional information. The algorithms and techniques employed in this toolbox are all openly available in literature. PEM and PEP do not employ any protected or patented material, but instead provides a foundation for building empirical prognostic models for fast prototyping and model comparison. Appendix A includes an example application of the PEM toolbox, discussed later, and PEP toolboxes for health monitoring of the turbofan engine data presented in the 2008 PHM Challenge.

16.1.1 System Requirements

The system requirements for the Toolboxes are based on the requirements of MATLAB 7.11, which was used for development. Although some functions will execute properly on previous releases of MATLAB, it is suggested that that the Toolboxes be run on MATLAB 7 to ensure all functions execute as intended.

16.1.2 Toolbox Requirements

The PEM and PEP Toolboxes employ functions from several existing MATLAB toolboxes. In order to use all functions, the following toolboxes must be available:

1) Neural Networks Toolbox release 4.0 or later
2) Wavelet Toolbox release 3.0 or later
3) Statistics Toolbox, release 7.4 or later
4) Signal Processing Toolbox, release 6.14 or later

5) G
lobal Optimization Toolbox, release 3.1 or later

16.1.3 Toolbox Architecture

The current architecture of the PEP Toolbox, shown in Figure 16-1, includes functions supporting each of the three types of prognostics, which are discussed in the next chapter. Each of these modeling algorithms includes methods for estimating the 95% uncertainty interval of the remaining useful life (RUL) estimates. Additional functionality is available to support model development. Each of the available functions and their use is discussed in later chapters.

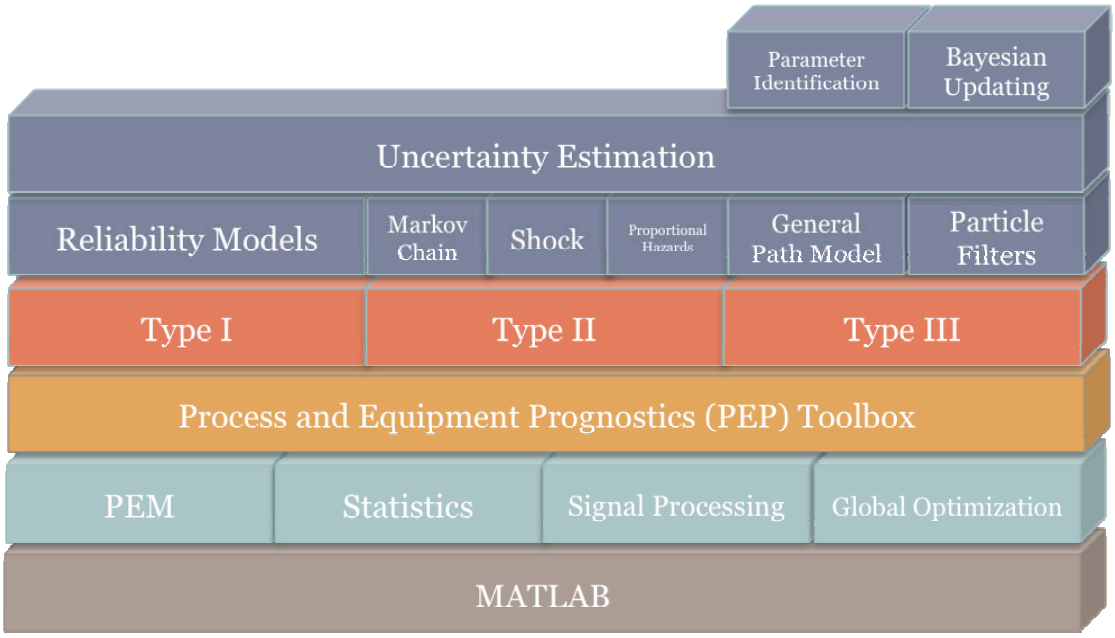


Figure 16-1: PEP Toolbox Architecture

16.1.4 Release Notes

This documentation has been adapted from the PEP version 1 users guide and PEM help files, as well as updates to the PEP since its initial release. The primary motivation for this revised version is as part of a contract between the University of Tennessee's Nuclear Engineering Department and the Nuclear Energy University Programs as part of the Department of Energy.

The guide includes background information on a complete prognostics paradigm, which includes monitoring, fault detection, and multiple prognostics technique depending on information available. It also includes Bayesian transitions between prognostic types.

The core functions of the PEP Toolbox facilitate conventional reliability models, Markov chain, shock, proportional hazards, general path, and particle filtering prognostic models. It also

includes methods for estimating the RUL uncertainty for each model type, commonly based on Monte Carlo procedures. Algorithms are included to identify prognostic parameters from multiple data sources for Type III models. All methods and algorithms employed in the PEP Toolbox are widely available in the open literature; as such, there is no alternate research version of the PEP Toolbox as there is in the PEM toolbox.

16.2 Background

Prognostics is one component of a complete health monitoring system which also includes system monitoring, fault detection, diagnostic modules, and operation and maintenance planning as shown in Figure . Full health monitoring systems, also called Condition Based Maintenance (CBM) systems, are the focus of much research. Data collected from a system of interest is monitored for deviations from normal behavior. Monitoring can be accomplished through a variety of methods, including first principle models, empirical models, and statistical analysis [Hines et al., 2006]. The monitoring module can be considered an error correction routine; the model gives its best estimate of the true value of the system variables. These estimates are compared to the data collected from the system to generate a time-series of residuals. Residuals characterize system deviations from normal behavior and can be used to determine if the system is operating in an abnormal state. In Figure below, the residuals have the same shape and fail at similar values thus they can be used as a prognostic parameter.

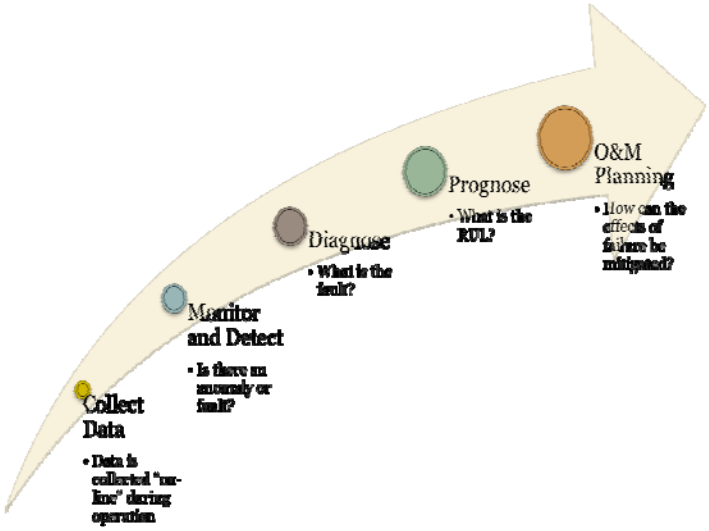


Figure 16-2: Complete Health Management System (Coble & Hines 2008)

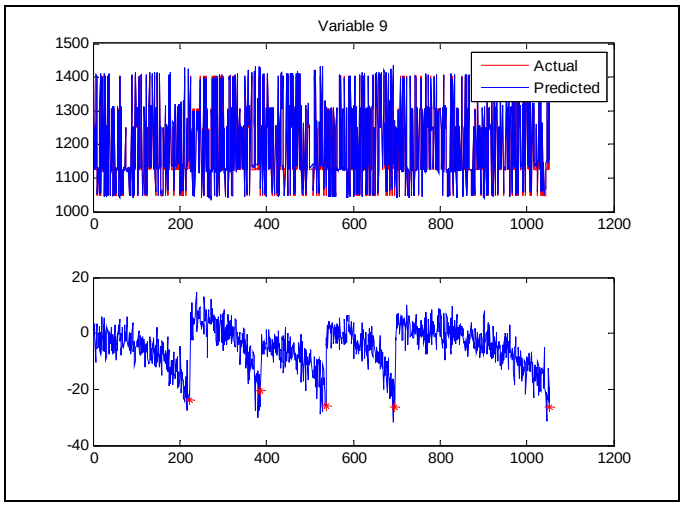


Figure 16-3: Example of Residuals used as a Prognostic Parameter

A common residual test for anomalous behavior is the Sequential Probability Ratio Test (SPRT) [Wald, 1945]. This statistical test considers a sequence of residuals and determines if they are more likely from the distribution that represents normal behavior or a faulted distribution, which may have a shifted mean value or altered standard deviation from the nominal distribution. If a fault is detected, it is often important to identify the type of fault; systems will likely degrade in different ways depending on the type of fault and so different prognostic models will be applicable. Expert systems, such as fuzzy rule-based systems, are common fault diagnosers. Finally, a prognostic model is employed to estimate the Remaining Useful Life (RUL) of the system or component. This model may include information from the original data, the monitoring system residuals, and the results of the fault detection and isolation routines. By applying the entire suite of modules, one can accomplish the goals of most prognostic systems: increased productivity; reduced downtime; reduced number and severity of failures, particularly unanticipated failures; optimized operating performance; extended operating periods between maintenance; reduced unnecessary planned maintenance; and reduced life-cycle cost.

Development of an integrated health management system can be daunting because of the high level of complexity involved in identifying appropriate algorithms at each stage. To support this, a suite of MATLAB - based toolboxes have been developed at the University of Tennessee PROaCT lab with a range of monitoring, fault detection and prognostic capabilities. The Process and Equipment Monitoring (PEM) toolbox was developed to facilitate auto - associative modeling of process and system data, and fault detection [2]. The PEM toolbox includes auto -

associative kernel regression models, auto-associative neural networks, and linear regression models for system monitoring and sequential probability ratio test and error uncertainty limit monitoring fault detection methods. The results of both of these modules, as well as an independently developed diagnostic module, if available, can be used to facilitate prognostic analysis. The Process and Equipment Prognostics (PEP) toolbox is a MATLAB ‑ based toolbox developed to aid in development of empirical prognostic models of each of the three types. The PEP toolbox is designed to integrate with the previously developed PEM Toolbox. The results of process monitoring and fault detection produced by the PEM toolbox as well as the original system data are utilized by the PEP toolbox to make RUL predictions for the system, as shown in Figure .

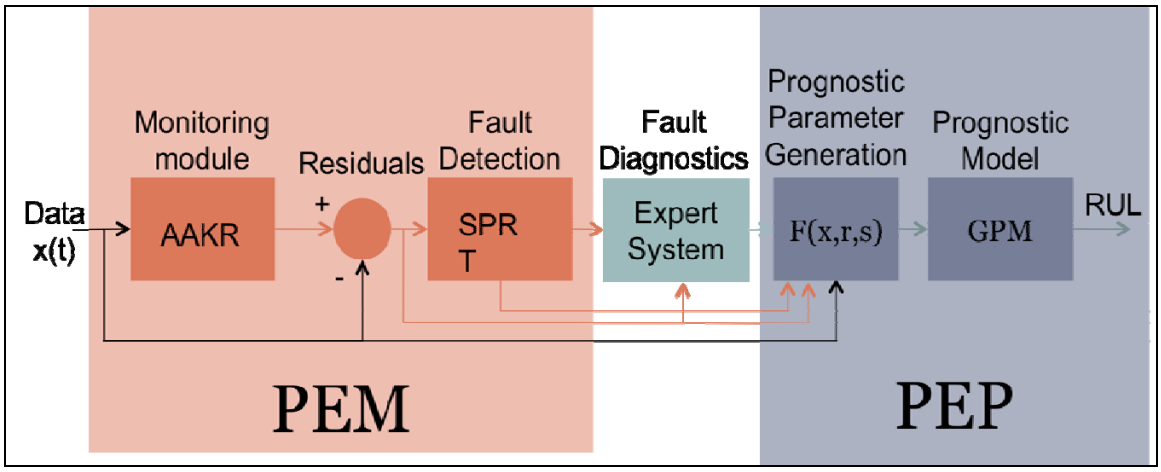


Figure 16-4: PEM and PEP Toolbox Flow Chart

The purpose of the PEP toolbox is to provide a base set of tools to facilitate prognostic model development. A myriad of prognostic algorithms have been developed which use a variety of information sources, models, data processing algorithms, etc. Typically, prognostic model development depends highly on the expertise of the developer. The PEP toolbox reduces the development burden on the system designer and facilitates the rapid development and quantitative performance characterization of competing models.

A variety of algorithms have been developed for application to specific systems or classes of systems. The efficacy of these algorithms for a new process depends on the type and quality of data available, the assumptions inherent in the algorithm, and the assumptions that can validly be made about the system. As such, these prognostic algorithms can be categorized according to many criteria. One proposed categorization is based on the type of information used to make prognostic estimates; this results in three classes of prognostic algorithms, as shown in Figure.

Type I prognostics is traditional time to failure reliability analysis; this type of prognostic algorithm characterizes the expected lifetime of an average system operating in a historically average environment. Type II methods characterize the average life under specific usage conditions. They can be used if operating condition stressors, such as load, input current and voltage, ambient temperature, vibration, etc., are measurable and correlated to system degradation. Algorithms in this class include specific formulations of the Markov Chain model, shock model, and proportional hazards model.[Hines, 2007] The final class of algorithms, Type III, or condition - based prognostics, characterizes the lifetime of a specific unit or system operating in its specific environment; these are the only truly individual-based prognostic models. These methods attempt to trend some measure of degradation, either directly measured from the system or inferred from other measurements, to a pre-defined failure threshold. The PEP toolbox includes algorithms representing each of the three categories. These algorithms are described in broad detail in the following sections. References to more detailed discussions are given for the interested reader.

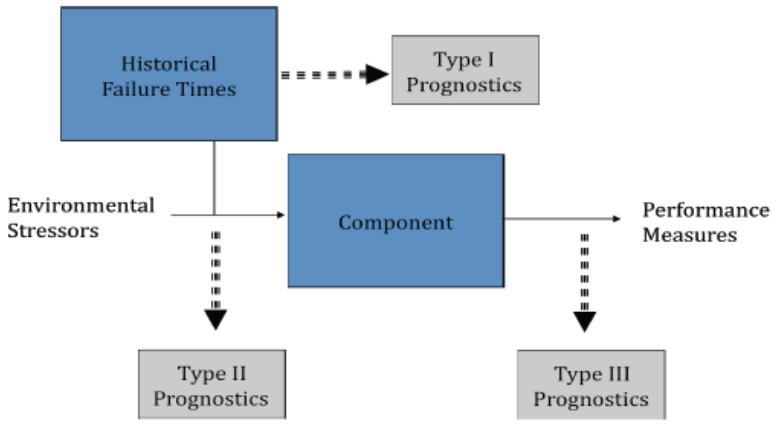


Figure 16-5: Prognostic Algorithm Categorization

16.2.1 Monitoring

Multiple monitoring techniques exist and are widely implemented for a plethora of reasons. They are useful for sensor calibration, condition monitoring, and are essential for fault detection and empirical prognostics. The Multivariate State Estimation Technique is one such proprietary technique used by Argonne National Laboratory [XX]. Other non-proprietary techniques are the Auto-Associative Neural Network (AANN) and the Auto-Associative Kernel Regression (AAKR). Auto-Associative refers to the access of data, using part of the data itself. This technique, while popularized through the use of AANNs, is valid in other applications.

The technical details of AANNs can easily be referenced in outside literature. In most cases, the key factor of AANN is the dimensional bottleneck between input and output. This bottleneck in the one or more hidden layers compresses information and can handle non-linear forms. The models are optimized usually by backpropagation of error based on training cases.

16.2.2 AAKR

The AAKR holds fault-free observation vectors, **x**, represented as a 1xk vector of k signals.

$$\boldsymbol{x} = [x_1 \quad x_2 \quad \cdots \quad x_k]$$

These vectors are stored in memory matrix **X** of n observations.

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,k} \end{bmatrix}$$

For every query vector input, **z**, into the AAKR, the distance between the query and each memory vector $\mathbf{x_i}$ is calculated based on the Euclidean distance.

$$d_i(x_i, z) = \sqrt{(x_{i,1} - z_1)^2 + (x_{i,2} - z_2)^2 + \cdots + (x_{i,k} - z_k)^2}$$

Calculating the distance for each memory vector results in a nx1 matrix of distances **d**. These distances are transformed into weights based on the Gaussian kernel of bandwidth h.

$$w = K_h(d) = \frac{1}{\sqrt{2\pi h^2}} e^{-d^2/h^2}$$

The expected output vector is then the weighted average of all memory vectors.

$$\hat{x} = \frac{\sum_{i=1}^{n}(w_i x_i)}{\sum_{i=1}^{n} w_i} = \frac{w^T X}{a}$$

If a is defined as the sum of all weights. Thus the two parameters to be optimized are the memory matrix and the kernel bandwidth.

16.2.3 Performance Metrics

Once the model of preference is built, several performance metrics may be examined that help to quantify how well the model performs. The first metric, dubbed accuracy, can be more technically described as mean squared error. For N test observations, it is calculated as follows:

$$A = \frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2$$

Because it is a measure of error, a lower value is desirable.

The next two metrics are closely related and are called auto-sensitivity and cross-sensitivity. They are also alternatively referred to as robustness and spillover [XX]. In both cases, they can be thought of intuitively as how much each sensor resists a drift. The auto-sensitivity measures how a single sensor resists a drift input of the same sensor. The cross-sensitivity for how much a drift bleeds into other sensors. For drifted prediction $\hat{x}^{drift}$, unfaulted prediction $\hat{x}$, drifted input x^{drift}, and unfaulted input x, the auto-sensitivity of each signal i, is

$$S_{A,i} = \frac{1}{N}\sum_{k=1}^{N}\left|\frac{\hat{x}_{ki}^{drift} - \hat{x}_{ki}}{x_{ki}^{drift} - x_{ki}}\right|$$

with the analogous cross-sensitivity as

$$S_{C,j} = \frac{1}{N}\sum_{k=1}^{N}\left|\frac{\hat{x}_{kj}^{drift} - \hat{x}_{kj}}{x_{ki}^{drift} - x_{ki}}\right|$$

where j is the index of un-faulted variable under scrutiny. In both cases a value closer to 0 is desirable. This means that they resist drifts, and can correctly fix the inputs, leading to residuals that resemble the drift in magnitude. For sensor calibration purposes, it correctly shows sensor drifts. For prognostics, the residuals contain degradation information.

16.2.4 Type I Prognostics

Type I methods are a simple extension of traditional reliability analysis, based entirely on an a priori distribution of failure times for similar systems in the past. Prognostic algorithms in this class characterize the average lifetime of an average system operating in historically average conditions; they do not utilize any information specific to the system at hand. The main assumption made when applying Type I methods is that future systems will operate under similar conditions to those seen in the past and will fail in similar ways.

Typically, Type I prognostic models track a population of systems over their lifetime and record only the failure time of each system. In addition, the total runtime of each system which hasn't failed at the end of the observation is recorded; this is called censored data and is also included in the analysis. A probability distribution is fit to these runtimes to give an estimate of the time of

failure (ToF) distribution of the population. The most common parametric model used in reliability analysis is the Weibull distribution. This model is used because it is flexible enough to model a variety of failure rates. The formula for the failure rate, λ(t), is a two parameter model with a shape parameter (β) and a characteristic life (θ):

$$\lambda(t) = \frac{\beta}{\theta}\left(\frac{t}{\theta}\right)^{\beta-1}$$

and the failure probability density is given by:

$$f(t) = \left(\frac{\beta}{\theta}\right)\left(\frac{t}{\theta}\right)^{\beta-1} e^{-\left(\frac{t}{\theta}\right)^{\beta}}$$

The two parameters in the Weibull model provide the modeling flexibility for components exhibiting an increasing failure rate (β>1), a constant failure rate (β=1), and a decreasing failure rate (β<1). With the correct choice of shape parameter, the Weibull distribution adequately models the exponential, normal, or Rayleigh distributions. Examples of different shape parameters are given in Figure , where f(t) is the probability density of the Weibull distribution. Additional information on Weibull modeling is available in Abernethy [1996].

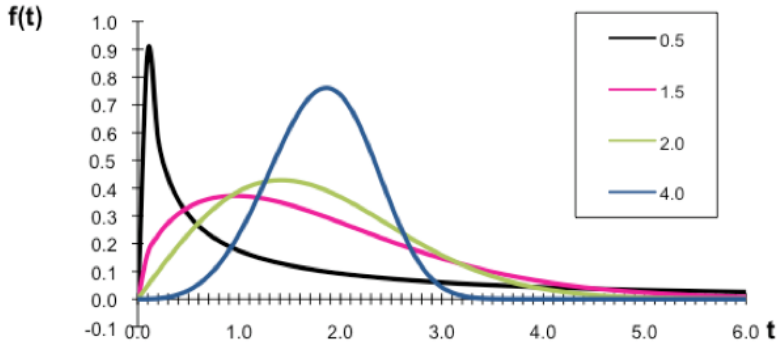


Figure 16-6: Weibull Failure Distributions with Different Shape Parameters

Most commonly, the Mean Residual Life (MRL) is used to estimate the RUL of a system using Type I prognostics. For a unit of age *t*, the MRL method assumes that the remaining life is a random variable, and the MRL is given by the expected value of this random variable [Guess and Proschan, 1985]:

$$MRL(t) = \frac{1}{S(t)}\int_{t}^{\infty} S(u)du$$

where S(·) is the survival function and t is the current time. The survival function is the complemet of the CDF and gives the probability of surviving beyond time t. The MRL at time *t* can be calculated from either parametric or nonparametric distributions, which makes it particularly flexible for application to real world data.

16.2.5 Type II Prognostics

Type II prognostic algorithms incorporate information about the operating conditions of the system into estimates of RUL. The PEP toolbox supports three Type II algorithms: Markov Chain models, shock models, and proportional hazards models.

16.2.6 Markov Chain Models

The Markov Chain model is based on the assumption that the next state which a system will occupy depends only on the current state; past states do not affect the probability of transitioning to a new state. There are two types of Markov Chain prognostic models, which vary only in the information they use to simulate possible future states. Type II Markov Chain models are composed of two models.

The first model, called the environmental model, is a Markov Chain simulation which produces possible future operating state progressions based on transition probabilities seen in the past and the current operating state. The environment model is needed for making a prediction as to how the environment and operating conditions evolve in the future. The environmental model is defined by the transition probability matrix, Q:

$$Q = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix}$$

where p_{ij} is the probability of transitioning from state *i* to state *j*. Often this probability matrix is assumed to be static, but it is straightforward to extend the method to time-dependent or degradation level-dependent transition probabilities. This model is used to simulate many possible future state progressions beginning at the current state.

These state progressions are then mapped to a degradation measure, which is the second model necessary in the Type II Markov Chain algorithm. The degradation measure is represented as a function of observable environmental conditions. To be useful for making a reliability prediction, the function should reflect the manner in which the environmental conditions affect the component reliability. Usually, environmental stressors tend to deteriorate the component

reliability in a cumulative manner. Hence, the function to relate the environment conditions to the prognostic parameter is commonly of a cumulative form:

$$Y(t_k) = \sum_{i=1}^{k} g\big(E(t_i, t_i + \Delta t_i)\big)\Delta t_i$$

where $Y(t_k)$ is the degradation measure value at time t_k, $E(t_i, t_i+\Delta t)$ is the environmental condition observed at the time interval $[t_i , t_i+\Delta t]$, and $g(.)$ is an appropriate function of environmental conditions.

When the estimated degradation measure is found to cross some pre-defined threshold, failure is said to have occurred. At each time of interest, many possible state progressions are simulated and mapped to degradation measures. These measures are then used to define a time of failure (ToF) distribution for the system. At each time of interest, many degradation paths are simulated, and a probability of failure distribution is estimated from the collection as shown in Figure.

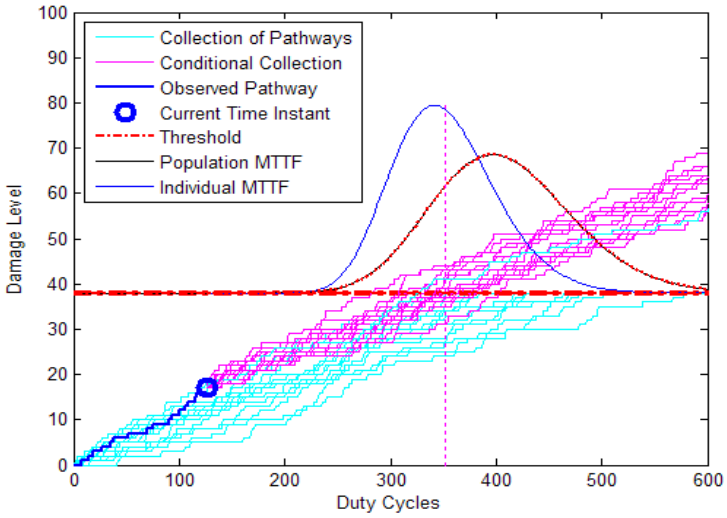


Figure16-7: Markov Chain Model PoF estimation

In Figure 6, the blue lines are a collection of degradation pathways that grow towards the failure threshold over time. If the actual degradation is measureable, indicated by a dark blue path, then the model can be used to simulate future pathways from the current state. These are represented by the purple paths. The collection of degradation paths can be used to predict the failure distribution. The red distribution represents the population failure distribution while the blue distribution is the predicted distribution for the individual. [Hines, 2007]

16.2.7 Shock Models

The Markov Chain model is continuous in the time domain, but discrete in the degradation measure. A more general formulation is the Shock model [Esary and Marshall, 1973; Gut, 1990; Mallor and Santos, 2003]. Instead of experiencing some known amount of shock at each random shock occurrence, the shock model allows for a shock of random size. Shock models have three parameters that are estimated from historical data: time between successive shocks, $t \sim \text{Exp}(\lambda)$, magnitude of the shocks, $x \sim F(x)$, and the critical failure threshold.

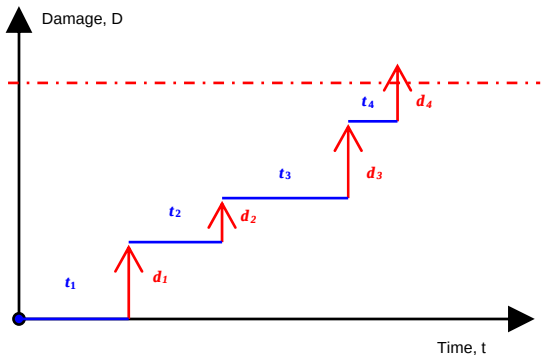


Figure 16-1: Shock Model Example

In this model, the time between random shocks is a continuous random variable, with the probability of shock often determined by the current degradation state, the operating conditions, or some combination thereof. The size of the shock may be based on a single shock size distribution, or other features such as the current degradation measure, the operating condition, or other measures available from the system may define it. Again, when the cumulative degradation measure crosses some pre-determined threshold, failure is said to have occurred; a probability of failure distribution is estimated from multiple simulated degradation measures.

16.2.8 Proportional Hazards Models

The Proportional Hazards (PH) Model developed by Cox [1984] merges failure time data and stress data to make RUL estimates. The basic proportional hazard model assumes that the observed hazard rate is separable into a baseline hazard rate dependent only on time, $\lambda_0(t)$, and a second function which is independent of time but dependent on operating conditions, called covariates, $\psi(z;\beta)$, where z is a numerical quantification of the covariates and β is regression parameters. The model uses the covariates to modify the baseline hazard rate to give a new hazard rate for the system's specific usage conditions:

$$\lambda(t;z) = \lambda_0(t)\exp\left(\sum_{j=1}^{q}\beta_j z_j\right)$$

Therefore the observed hazard rate at any given covariate condition is always a multiple of the baseline hazard rate, $\lambda_0(t)$, and a proportionality constant determined by the covariate function, $\psi(z;\beta)$, hence the name proportional hazards model. Generally $\psi(z;\beta)$ is assumed to have some type of known functional form such as logistic, linear, inverse linear, or exponential. Typically, and in the PEP Toolbox, this function is assumed to be exponential. Generally $\lambda_0(t)$ does not have any assumed functional form, and the PEP Toolbox utilizes a non-parametric, empirical-based baseline hazard rate.

Failure data collected at a variety of covariate operating conditions are used to solve for the parameters (β_j) using an ordinary least squares algorithm. The baseline hazard is the hazard rate when covariates have little or no influence on the failure rate. A basic assumption of the proportional hazards model is that the effects of these covariates are multiplicative; this means that when the ratio of two covariates is evaluated, their hazard rates are proportional.

In the following example, a tire is operated in one of three operating conditions: normal, off-road, and high slip. To check for proportionality of the covariates of all three operating conditions, the log of the negative log of the reliability function is plotted as seen in Figure below.

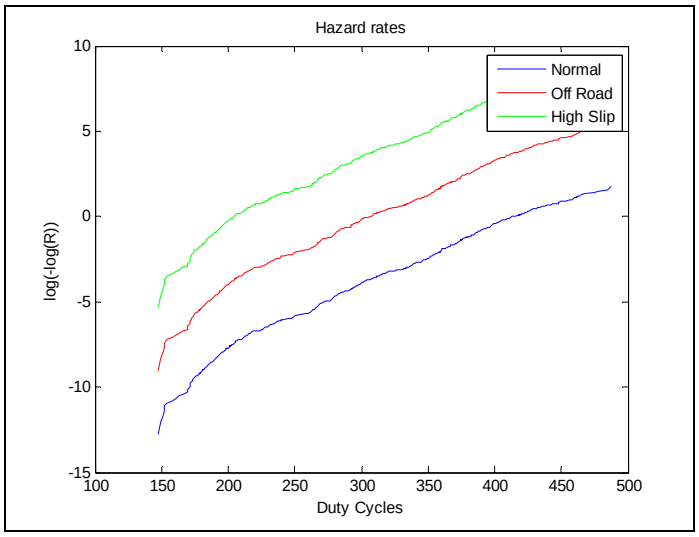


Figure 16-9 Check for Proportionality of PHM Covariates

The baseline hazard rate is then estimated with the covariate value of 0 for normal conditions. The predicted hazard rate can be seen in Figure 16- below.

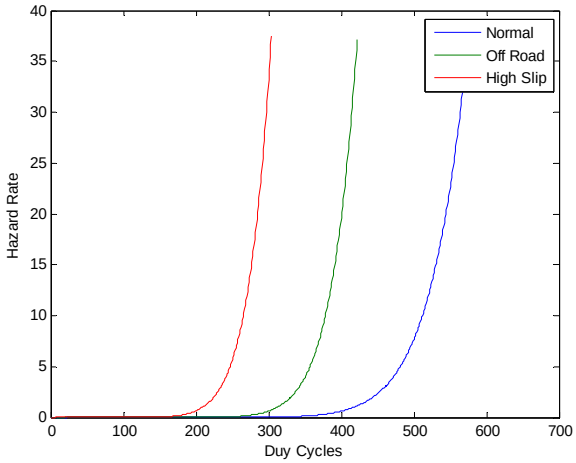


Figure 16-10 Estimated Cumulative Hazard Rates

Finally, the samples are used to collect the PH model's coefficients with the covariates of 1 for off road and 2 for high slip. The model is then run and results in a regression coefficient that can then be used to model the hazard rates. Figure 16- below shows the comparison of the actual data to the PH model's results. A full discussion of developing a proportional hazards model can be found in [Kumar and Kelfjo, 1994].

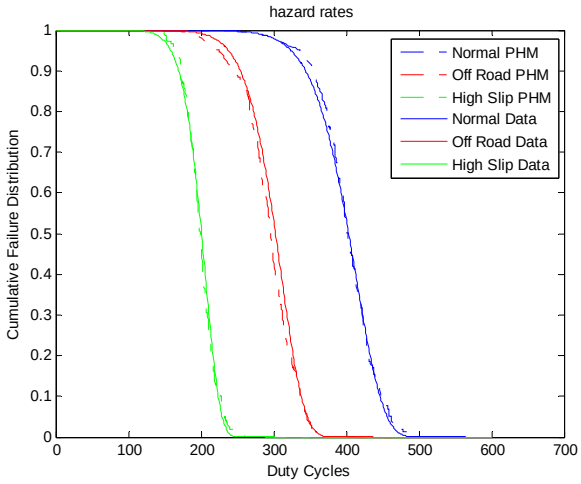


Figure 16-11 Comparison of Actual Data to PH Model's Results

16.2.9 Type III Prognostics

Type III models consider the actual condition of the system, either directly measured from the system or inferred from other measurements. These condition measurements, often called

degradation measures or prognostic parameters, are used to extrapolate from the current condition to a pre-defined critical failure threshold. The following section introduces the idea of the prognostic parameter. This is followed by descriptions the two Type III models available in the PEP Toolbox: general path model and particle filtering model. Finally, a short discussion of selecting an appropriate prognostic parameter is given, which necessary for either algorithm.

16.2.10 Prognostic Parameter

Effects-based prognostics uses degradation measures to form a prognostic prediction. A prognostic parameter, also called a degradation measure, is a scalar or vector quantity that numerically reflects the current ability of the system to perform its designated functions properly. It is a quantity that is correlated with the probability of failure at a given moment. A degradation path is a trajectory along which the degradation measure is evolving in time towards the critical level corresponding to a failure event. Type III prognostics attempt to extrapolate along this degradation path to determine the RUL of a component or system.

The degradation measure does not have to be a directly measured parameter. It could be a function of several measured variables that provide a quantitative measure of degradation. It could also be an empirical model prediction of the degradation that cannot be measured. For example, pipe wall thickness may be an appropriate degradation parameter but there may not be an unobtrusive method to directly measure it. However, there may be related measurable variables that can be used to predict the wall thickness. In this case the degradation parameter is not a directly measurable parameter but a function of several measurable parameters. Monitoring system residuals are intuitive candidates for prognostic parameters because they naturally characterize how "far" a system is from normal operation.

When the degradation level of a system reaches some predefined critical failure threshold, the system is said to have experienced a soft failure; for example, car tire tread is below some specified depth. These failures generally do not concur with complete loss of functionality, as in a hard failure; however, they correspond with the time when an operator is no longer confident that equipment will continue to work to its specifications. Both general path models and particle filters attempt to extrapolate the prognostic parameter to a critical failure threshold to estimate the RUL; these algorithms are described in the following sections.

16.2.11 General Path Model

The General Path Model (GPM) was first proposed by Lu and Meeker [1993] to move reliability analysis from failure time to failure mode analysis. The GPM reliability methodology has a natural extension to estimation of RUL of an individual component or system. GPM analysis

begins with some assumption of an underlying functional form of the degradation path for a specific fault mode. The degradation of the i^{th} unit at time t_j is given by:

$$y_{ij} = \eta(t_j, \phi, \theta_i) + \varepsilon_{ij}$$

where φ is a vector of fixed (population) effects, θ_i is a vector of random (individual) effects for the i^{th} component, and $\varepsilon_{ij} \sim N(0,\sigma^2_\varepsilon)$ is the standard measurement error term. The model parameters are estimated from the available data. This degradation path model, y_i, can be extrapolated to the failure threshold, D, to estimate the component's time of failure.

As data is collected during use, the degradation model is fit for the individual component. This specific model can be used to project a time of failure for the component. Because of noise in the degradation signal and uncertainty in the failure threshold, the projected time of failure is not exact. Monte Carlo techniques are used with estimates of the uncertainty sources to project a 95% uncertainty interval around the RUL estimate.

The traditional GPM methodology considers only the data collected on the current unit to fit the degradation model. However, prior information is available from the historic degradation paths used for initial model fitting, including the mean degradation path and associated distributions. This data can provide valuable knowledge for fitting the degradation model of an individual component, particularly when only a few data points have been collected or the collected data suffers from excessive noise. Bayesian updating methods are used to incorporate this additional information in the fitted model.

Bayesian updating is a method for combining prior information about the set of model parameters with new data observations to give a posterior distribution of the model parameters (Figure 16-). This allows both current observation and past knowledge to be considered in model fitting.

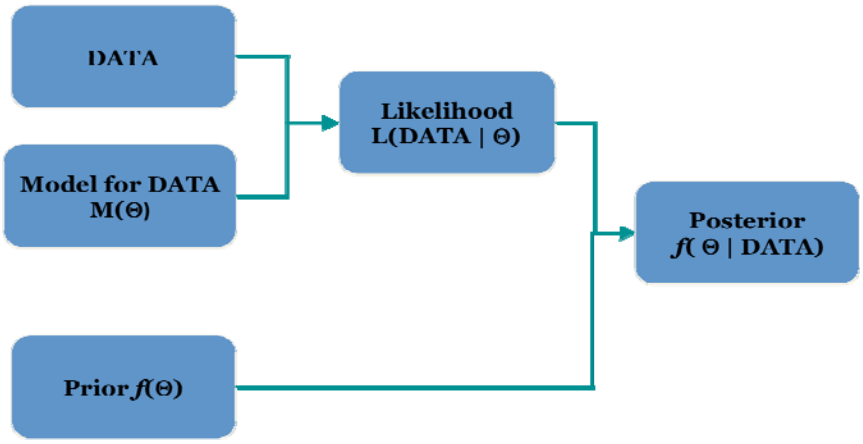


Figure 16-12: Bayesian Updating Methodology

The PEP Toolbox allows for Bayesian updating of linear regression models. These models are not necessarily linear models, but are linear-in-parameters. In this vein, a linear regression model is given by:

$$Y = bX$$

The model parameters are estimated as:

$$b = \left(X^T \Sigma_y^{-1} X\right)^{-1} X^T \Sigma_y^{-1} Y$$

where Σ_y is the variance-covariance noise matrix for the response observations. It is important to note that the linear regression model is not necessarily a linear model. The data matrix X can be populated with any function of degradation measures, including higher order terms, interaction terms, and functions such as $sin(x)$ or e^x. If prior information is available for a specific model parameter, i.e. $\beta_j \sim N(\beta_{jo}, \sigma^2_\beta)$, then the matrix X should be appended with an additional row with value one at the j^{th} position and zero elsewhere, and the Y matrix should be appended with the a priori value of the j^{th} parameter.

$$X^* = [\ X;\quad 0\ \cdots\ 0\ \ 1\ \ 0\ \cdots\ 0\ ]$$
$$Y^* = [\ Y;\quad \beta_j\ ]$$

Finally, the variance-covariance matrix is augmented with a final row and column of zeros, with the variance of the a priori information in the diagonal element.

$$\Sigma_y^* = \begin{bmatrix} \sigma_y^2 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & \vdots \\ 0 & \cdots & \sigma_y^2 & 0 \\ 0 & \cdots & 0 & \sigma_{\beta_j}^2 \end{bmatrix}$$

If knowledge is available about multiple regression parameters, the matrices should be appended multiple times with one additional row for each parameter.

It is convenient to assume that the noise in the degradation measurements is constant and uncorrelated. Some a priori knowledge of the noise variance is available from the exemplar degradation paths. If this assumption is not valid for a particular problem, then other methods of estimating the noise variance must be used. The assumption of uncorrelated noise allows the variance-covariance matrix to be a diagonal matrix consisting of noise variance estimates and a priori knowledge variance estimates. If this assumption is not valid, including covariance terms is trivial; again these terms can be estimated from historical degradation paths.

After a priori knowledge is used to obtain a posterior estimate of degradation parameters, this estimate becomes the new prior distribution for the next estimation of degradation parameters. The variance of this new knowledge is estimated as:

$$\frac{1}{\sigma^2_{posterior\beta_j}} = \frac{n}{\sigma_y^2} + \frac{1}{\sigma^2_{prior\beta_j}}$$

where n is the number of observations used to fit the current model.

Bayesian updating is particularly effective when few data points are available or the data is contaminated with a high level of noise. An example application is given here to illustrate the efficacy of Bayesian techniques. Figure 16- shows the prognostic parameter for a population of failed systems. These parameters have a clear, negative trend toward failure which is roughly quadratic in nature. For this GPM system, a quadratic function is fit to data from a new system and extrapolated to a critical failure threshold of approximately -20. Figure 16- shows the results of applying the GPM to approximately 90 observations of noisy data from a new system. As the figure shows, the small amount of data available includes noise levels which preclude appropriate model fitting. Figure 16- shows the prognostic result for the same system when Bayesian updating is employed. In this model, the inadequate data available from the system is augmented with prior information to "force" the model fit to take the downward shape which is historically expected. The RUL estimate obtained with the Bayesian approach is 135 cycles, versus an

undeterminable estimate obtained from the non-Bayesian approach. The actual RUL after the first 84 observations is 170 cycles, resulting in an RUL error of approximately 20%. While this error is still high, it is within a reasonable accuracy considering the amount of data available and will improve further as more data is collected.

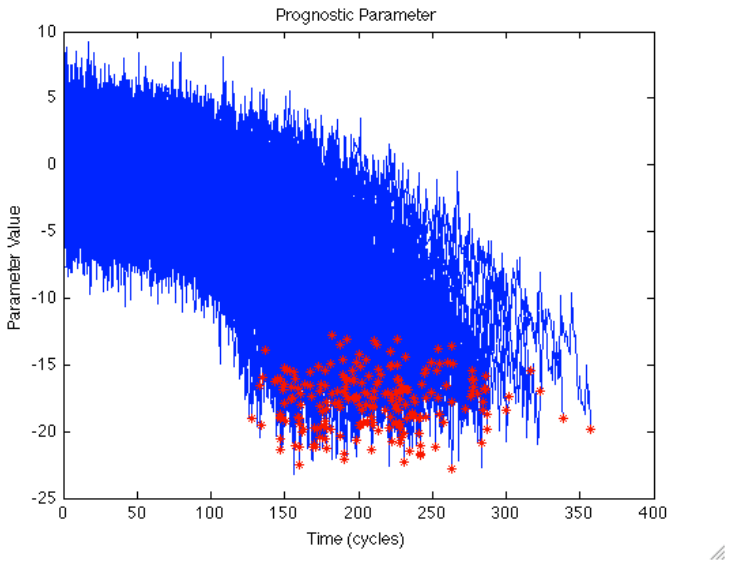


Figure 16-13: Prognostic Parameter for a Population of Failed Systems

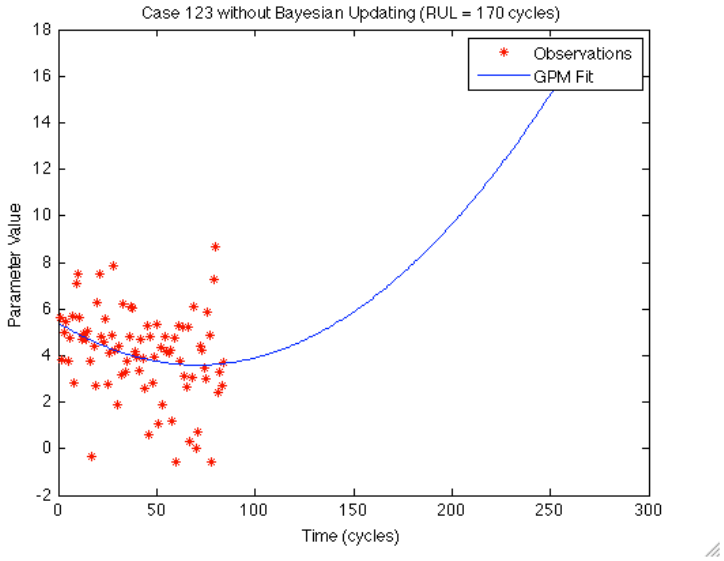


Figure 16-14: GPM fit without Bayesian updating

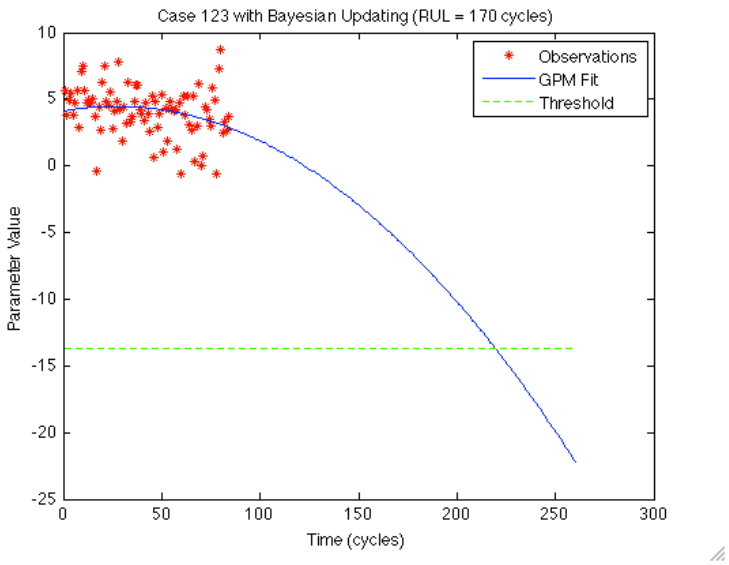


Figure 16-15: GPM fit with Bayesian updating

16.2.12 Particle Filtering Model

Particle filtering was originally developed to provide an estimation of the marginal probability in Bayes' Theorem that would allow for modeling of nonlinear systems and potentially non-Gaussian noise [Cadini, 2009]. The particle filter method utilizes Monte Carlo simulation to provide an approximate solution to the marginal distribution by generating artificial random samples and comparing their distribution to that of the measurements. The particle filter method first starts with Bayes' Theorem:

$$p(x_k|z_{0:k}) = \frac{p(z_k|x_k) * p(x_k|z_{0:k-1})}{p(z_k|z_{0:k-1})}.$$

Here, **x** represents the state-space vector of the system, which is not directly measured. In condition monitoring, this is typically the prognostic parameter or other measure of system health. The term **z** represents the vector of measurements. Measurements are first taken at time = 0; the "current" time, or the time of interest is represented by the subscript *k*, and the previous time step is *k-1*. The term $p(x_k|z_{0:k})$ is defined as the posterior distribution and represents the distribution of the likelihood of a system state $\mathbf{x}_k$ existing given the measurements $\mathbf{z}_{0:k}$ (i.e. all measurements, including the current measurement). The term $p(z_k|x_k)$ is the conditional probability and represents the likelihood that a given state would yield the current measurements. The term $p(x_k|z_{0:k-1})$ is the prior distribution and is the likelihood that a given state would exist based on all measurements prior to the current measurement. Finally, $p(z_k|z_{0:k-1})$ is the

marginal probability and represents the likelihood that the current measurements would occur given all previous measurements.

The long-standing difficulty with Bayes' Theorem is determining the marginal probability, and this is the purpose of particle filtering. The first step in particle filtering is Sequential Importance Sampling (SIS), where a known distribution is used to generate random samples for $\mathbf{x}_{0:k}$. (In contrast to GPM, SIS updates all particles simultaneously at a given time step rather than updating a single particle all the way to failure.) The distribution only needs to ensure that the range of possibilities is covered, though a distribution that closely resembles the true distribution of probabilities should provide faster convergence and more reliable results. This distribution is defined as an *importance function*:

$$q(x_k|z_{0:k}).$$

Next, weights for the sample particles are defined by relating the importance function to the posterior distribution:

$$w_k^i \propto \frac{p\left(x_k^i|z_{0:k}\right)}{q\left(x_k^i|z_{0:k}\right)}.$$

In the above equation, both terms in the ratio are unknown. However, the weights from the previous time step are known and can be used to approximate the weights according to (1). The approximation may require normalization to form a true pdf; this can be readily performed after all weights are estimated.

$$w_k^i \propto w_{k-1}^i * \frac{p\left(z_k|x_k^i\right)p\left(x_k^i|x_{k-1}^i\right)}{q\left(x_k^i|x_{0:k-1}^i, z_{0:k}\right)} \tag{1}$$

A visual example of the weighting process may be seen in Figure 16- and Figure 16-. Prior to weighting, all samples have equal weight. At time step 20, those particles whose states have a higher likelihood of representing the true state of the system (as estimated through measurements) receive greater weight; the particles with less likelihood of representing the system receive less weight. Having re-weighted the particles, they now represent an updated posterior distribution of the marginal probability and the new posterior distribution may be estimated.

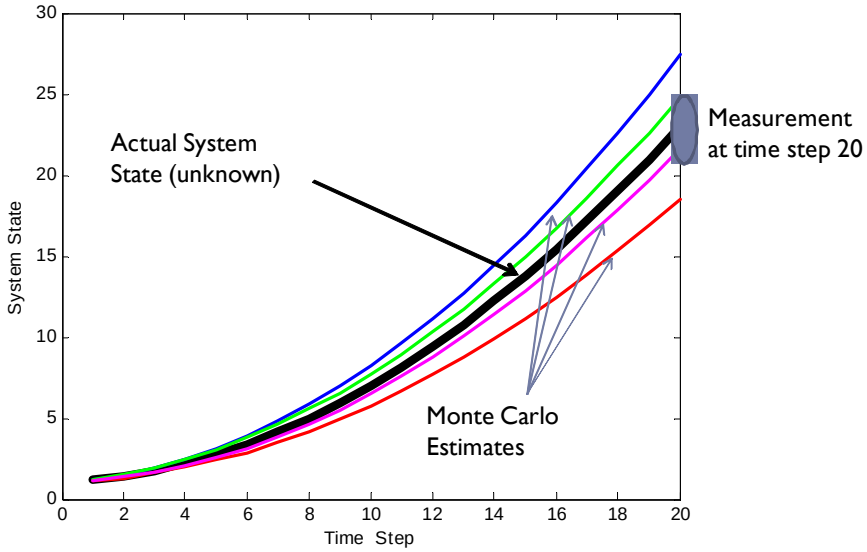


Figure 16-16: Sample Estimates of the System State Prior to Weighting

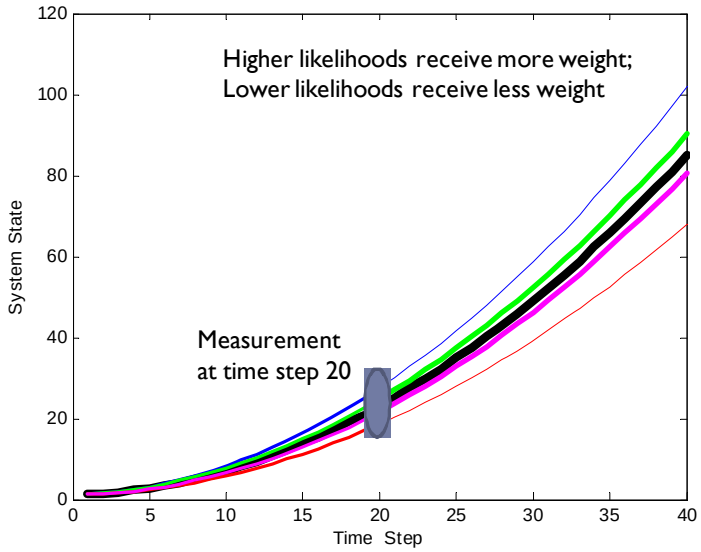


Figure 16-17: Sample Estimates of the System State after Weighting

As this process is repeated, the particle weights are continually updated every time Bayes' Rule is applied. Since SIS will add weight to particles with the highest likelihood at the expense of lower-likelihood particles, the process will eventually drive the weights of all particles to zero except for the highest-likelihood particle, a problem known as degeneracy. To avoid degeneracy, the particles are occasionally redistributed by Sequential Importance Resampling (SIR).

SIR may be conducted by a variety of methods, but the general approach is to replace the existing weighted particles with new unweighted particles chosen by the posterior distribution

provided by SIS. In review, SIS may be seen as a filter of particle weights where the weights are updated to fit the estimated posterior distribution. SIR may therefore be seen as a filter of the particles themselves, where new particles are chosen based on the weighted particles after SIS. This two-stage process is the heart of particle filtering.

Because particle filtering employs a Monte Carlo process, uncertainty estimates may be readily provided by the existing particle distributions. The state estimate at the present time is given by (2). The failure probability estimate at a future time $k+i$ is given by (3). Finally, the failure time distribution at a future time $k+i$ is given by (4).

$$p(x_k|z_{0:k}) \approx \sum_{i=1}^{N} w_k^i \delta(x_{0:k} - x_{0:k}^i) \tag{2}$$

$$\hat{p}(k+i) \approx \frac{\sum_{m,x_k^m \in \Omega_F} w_k^m}{\sum_{n,x_k^n \in \Omega^*} w_k^n} \tag{3}$$

$$p(t_k|z_{0:k}) \approx \sum_{i=1}^{N} w_k^i \delta(t_k - t^i) \tag{4}$$

Additional statistical inferences may be made via the particle distributions, such as 95% confidence intervals, hypothesis tests, etc.

Both Type III prognostic algorithms described rely on an appropriate prognostic parameter for modeling and extrapolating to failure. The PEP toolbox includes functionality to automatically select an appropriate, near-optimal prognostic parameter from available data sources. The following section briefly describes that process.

16.2.13 Selecting a Prognostic Parameter

An ideal prognostic parameter has three key qualities: monotonicity, prognosability, and trendability [Coble and Hines, 2009]. Monotonicity characterizes the underlying positive or negative trend of the parameter. This is an important feature of a prognostic parameter because it is generally assumed that systems do not undergo self-healing, which would be indicated by a non-monotonic parameter. However, this assumption is not valid for some components such as batteries, which may experience some degree of self-repair during short periods of nonuse. The monotonic trend is considered valid when considering an entire system, even if individual components or sub-systems may experience some self-repair. Prognosability gives a measure of the variance in the critical failure value of a population of systems. Ideally, failure should occur at

a crisp, well-defined degradation level. A wide spread in critical failure values can make it difficult to accurately extrapolate a prognostic parameter to failure. Finally, trendability indicates the degree to which the parameters of a population of systems have the same underlying shape and can be described by the same functional form. These three intuitive metrics are formalized in [Coble, 2010] to give a quantitative measure of prognostic parameter suitability.

Monotonicity is a straightforward measure given by:

$$Monotonicity = mean\left(\frac{\#pos\ \frac{d}{dx}}{n-1} - \frac{\#neg\ \frac{d}{dx}}{n-1}\right)$$

where n is the number of observations in a particular history. The monotonicity of a population of parameters is given by the average difference of the fraction of positive and negative derivatives for each path. When using data collected or inferred from actual systems, it is important to adequately smooth the data to give more accurate estimates of the derivatives. Numerical calculation of a function derivative should rarely be left to a simple difference function; the addition of noise makes this method inaccurate and impractical. In practice, fitting a line to a small portion of the data, perhaps five or ten observations, and taking the derivative to be the slope of that line will give a more realistic measure of the slope. When this method is employed, the above equation for monotonicity need only be adjusted for the number of calculated derivatives. Instead of n-1 derivatives, n-m derivatives may be calculated, where m is the number of data points used to calculate one derivative. It is important to note that the current formulation of monotonicity does not consider if the entire population is monotonic in the same direction, only that each individual exhibits an either generally increasing or decreasing trend. This is an undesirable feature in the prognostic parameter; however, it is considered in characterizing the prognosability which looks at how well clustered failure values are. If failure values for the entire population are well clustered and the individual parameters are monotonic, then the population must have either an increasing or decreasing monotonicity.

Prognosability is calculated as the deviation of the final failure values for each path divided by the mean range of the path. This is exponentially weighted to give the desired zero to one scale:

$$Prognosability = exp\left(-\frac{std(failurevalues)}{mean(|failurevalue - startingvalue|)}\right)$$

This measure encourages well-clustered failure values, i.e. small standard deviation of failure values, and large parameter ranges. This gives the model a long range to predict a very precise

value, which can be related to the notice period discussed previously. The failure values for the good prognostic parameter are very well clustered, following a wide range; the prognosability is 0.930. The failure values of the population of bad prognostic parameters cover a wide range of values; this parameter has prognosability of only 0.346.

Characterizing the trendability of a population of parameters poses significant difficulty compared to the other two metrics. A candidate parameter is trendable if the same underlying functional form can model each parameter in the population. Initially, trendability was characterized by comparing the fraction of positive first and second derivatives in each parameter. However, this naïve approach was highly susceptible to noise and did not provide a clear distinction between trendable and not-trendable parameters. An improved method for characterizing trendability is used in which prognostic parameters are re-sampled with respect to the fraction of total lifetime. This results in each prognostic parameter containing exactly 100 observations, with each observation corresponding to 1% of lifetime. The linear correlation is calculated across the population of prognostic parameters, and the trendability is given by the smallest absolute correlation:

$$Trendability = \min_{i=1m, j=1m}(|corrcoef_{i,j}|)$$

The PEP toolbox utilizes these suitability metric with genetic algorithm optimization to identify a near-optimal prognostic parameter from the available data sources. The prognostic parameter is optimized according to the following fitness function:

$$fitness = w_m monotonicity + w_p prognosability + w_t trendability$$

The weights may be adjusted to give more importance to certain metrics over others.

16.2.14 Bayesian Transitions

Each prognostic type takes in different information and requires different algorithms and models to apply. The data, and thus type, can be categorized chronologically to the component's life,. Type I before first startup and on. Type II before a fault is found. Type III to track the fault to failure. However there is not yet a unified approach that follows a component through its life, nor is there a way of preserving information from one type and carrying it to the next stage of analysis.

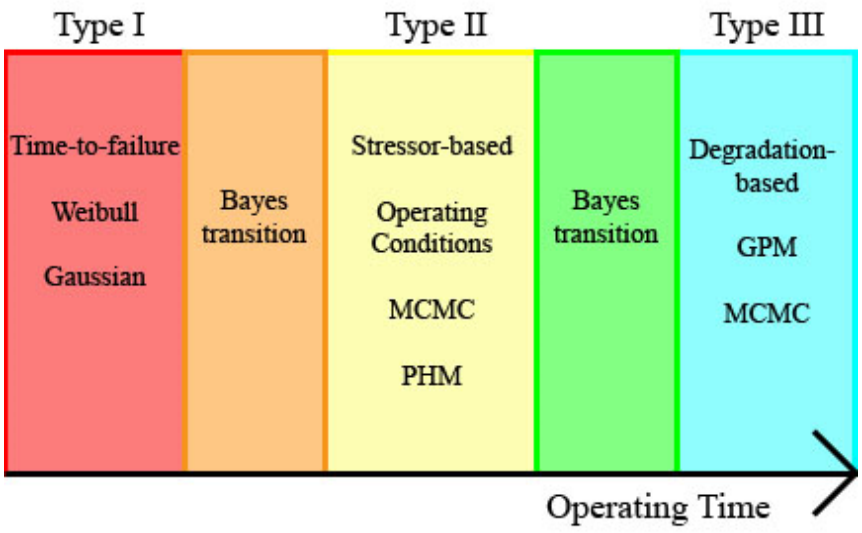


Figure 16-18 Lifecycle Prognostics with Bayes Transitions

One approach has been to apply Bayesian transitions. These methods combine previous estimates of a parameter, priors, with sampling data to produce a posterior estimate that combines both sources of data, while reducing uncertainty. This can be useful when investigating Lifecycle Prognostics. Bayesian transitioning methods can be applied to bridge between prognostics types.

16.2.15 Type I to Type II Transition

For this transition, Bayes formula can be applied using conjugate distributions. This is a widely practiced form of Bayes formula, and is characterized by relatively straightforward equations that are easily referenced. For example, the Gaussian conjugate distribution not only has Gaussian prior and posterior distributions, but also a normally distributed data sample.

First the Gaussian prior distribution is defined as

$$\pi(\mu,\tau) = \frac{1}{\sqrt{2\pi\tau^2}}\exp\left(\frac{-(\mu-\eta)^2}{2\tau^2}\right)$$

Let $X_1, X_2 \ldots X_n$ be independently and identically distributed (i.i.d.) sampling data ~ $N(\mu, \sigma^2)$.

$$f(x|\mu) = \prod_{i=1}^{n} f(x_i|\mu) = (2\pi\sigma^2)^{-n/2}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right)$$

These equations are combined with Equation XX to yield

$$E(\mu|x) = \frac{\frac{\eta}{\tau^2} + \frac{\sum x_i}{\sigma^2}}{\frac{1}{\tau^2} + \frac{n}{\sigma^2}} = \frac{\frac{\eta}{\tau^2} + \frac{n}{\sigma^2}\bar{X}}{\frac{1}{\tau^2} + \frac{n}{\sigma^2}}$$

$$var(\mu|x) = \frac{1}{\frac{1}{\tau^2} + \frac{n}{\sigma^2}}$$

Equation XX gives the maximum likelihood estimate (MLE) of the mean of the distribution, while equation XX gives the variance of the mean. It is important to distinguish between variance of the distribution, and variance of the mean. This variance is a measure of certainty of the expected mean estimate. The smaller the variance, the more precise the estimate is.

The prior variance τ^2 measures the strength of the belief in the uncertainty of the prior distribution. In this sense $1/\tau^2$ is the precision of the prior, while n/σ^2 is the precision of n data points. This means that the posterior mean is the weighted average of the prior and sample means with the precisions of each as weights. It also means that with more data, the prior is swamped out, and eventually the MLE approaches the posterior.

When applied to prognostics, the Type I RUL estimate distribution can be considered as the prior, with the Type II estimate as the sampled data. The fact that most Type I models use a Weibull distribution instead of a Gaussian can be a possible limitation. There does not exist straightforward solutions to a Weibull prior, as there is for the Gaussian. However, there are equations that re-paramaterize the Weibull parameters, transforming them into the mean and variance parameters of the Gaussian. This re-parameterization can retain much of the original distribution, especially if the distribution is similar to a Gaussian in the first place.

16.2.16 TypeI/II to Type III Transition

Bayesian priors can also be incorporated into the OLS model to reduce the uncertainty and increase the stability of RUL estimates. Bayesian statistics combines prior distributions with sampling data to create a posterior distribution. When very few data points are available the model can easily be thrown off and give wildly varying time of failures. When applied to OLS, the prior parameters, from all the failed cases, form the prior distribution. The sampled data comes from the censored data.

Using equation XX, it is then assumed that the parameters are normally distributed, allowing the use of the Gaussian conjugate distribution. The conditional posterior distribution is

then equation 13. Equations 14-16, based on the pseudoinverse solution equation 7, are used to solve equation 13, where n is the number of data points, and k is the degree of the polynomial fit.

$$\beta|\sigma^{\wedge}2, y \sim N(\beta^{\wedge}, V\sigma^{\wedge}2)$$

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

$$V = (X^T X)^{-1}$$

$$\sigma^2 = \frac{1}{n-k}(y - X\hat{\beta})^T (y - X\hat{\beta})$$

Before the Bayes prior information is incorporated, a data covariance matrix Σ is introduced. Instead of assuming equally distributed errors, $\sigma^2\mathbf{I}$, the covariance matrix is an n x n symmetric positive matrix. The previous equations are then replaced by their analogous versions.

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y$$

$$V\sigma^2 = (X^T \Sigma^{-1} X)^{-1}$$

$$\sigma^2 = \frac{1}{n-k}(y - X\hat{\beta})^T \Sigma^{-1} (y - X\hat{\beta})$$

To include Bayesian updating[13], the prior distribution of the path parameters ~$N(b_0,\Sigma_b)$ is treated as additional data points to the OLS solution. To achieve this, each variable is appended with the prior distribution data, equation 20. The **X** is appended with an identity matrix, with ones representing the parameters for which prior distributions exist. Y is appended with the prior mean values of the parameters. The diagonal matrix Σ is expanded with the variances of the prior parameters.

$$y_* = \begin{bmatrix} y \\ \beta_0 \end{bmatrix}, X_* = \begin{bmatrix} X \\ I_k \end{bmatrix}, \Sigma_* = \begin{bmatrix} \Sigma_y & 0 \\ 0 & \Sigma_b \end{bmatrix}$$

Another method, a novel approach presented in this research, is to treat a prior RUL distribution as an additional data point.

$$y_* = \begin{bmatrix} y \\ thresh \end{bmatrix}, X_* = \begin{bmatrix} X \\ MTTF \end{bmatrix}, \Sigma_* = \begin{bmatrix} \Sigma_y & 0 \\ 0 & \Sigma_{RUL} \end{bmatrix}$$

For example, if a Type I RUL distribution exists then the y is appended with the degradation threshold, **X** with the mean time to failure based on Type I analysis. And the diagonal matrix is appended with a measure of the RUL uncertainty distribution.

For both cases, the weight of the prior then depends on two main factors: the variance of the prior against the variance of the data, and the number of samples taken in. If the variance of the prior is small against the noise of the data, the prior b_0 will be weighed more heavily. However, no matter the difference in variance, with enough sampling, the data will eventually swamp out the prior in calculating the posterior.

16.3 PEM Functions List

The following is a partial list of functions from the PEM toolbox likely to be accessed directly by the user. While the PEM contains many more useful functions, a large portion are called by the more user-friendly high level functions provided here. Core low level functions are also provided to allow for the user to bypass PEM model architectures.

aagroup	Groups variables for use in an auto-associative empirical model
aakr	Low level implementation of auto-associative kernel regression
aars	Smoothes multivariate data using an auto-associative regression model
amfdet	Detects faults in a set of query data according to automated methods
autogroup	Automated variable grouping
cleandata	Cleans a set of data
confh	Consolidate fault hypothesis
eulm	Performs error uncertainty limit monitoring fault detection
fdetect	Detect faults within a set of data
gcplot	Creates a plot of the absolute correlation coefficients
initmodel	Initializes a PEM toolbox model structure
kr	Low level implementation of kernel regression
modchar	Characterizes an empirical model
optmodel	Optimizes the architecture of an empirical model
runmodel	Runs a simulation of an empirical model for query data
setmsa	Sets one or more of a model's attributes
sprtn	Uses the SPRT for a multivariate normal distribution for fault detection
uicm	Performs uncertainty interval coverage monitoring fault detection
unscore	Un-scales mean center, unit variance of scaled data

vectsel	Selects representative vectors from a set of data
vensample	Performs a Venetian blind sampling of a set of data
zscore1	Scales data to have mean center, unit variance

PEM Functions Descriptions

aagroup

Autoassociative variable grouping

Syntax

```
[Groups Ungrouped Removed] = AAGROUP(X,CutOff)

[Groups Ungrouped Removed] = AAGROUP(X)
```

Description

This function groups variables for use in an autoassociative empirical model.

[Groups Ungrouped Removed] = AAGROUP(X,CutOff) performs a variable grouping of X by comparing correlations to CutOff. The result is a cell array of the selected variable indices for each group Groups, a matrix of the ungrouped sensor variables Ungrouped, and a matrix of variable indices for the removed constants Removed.

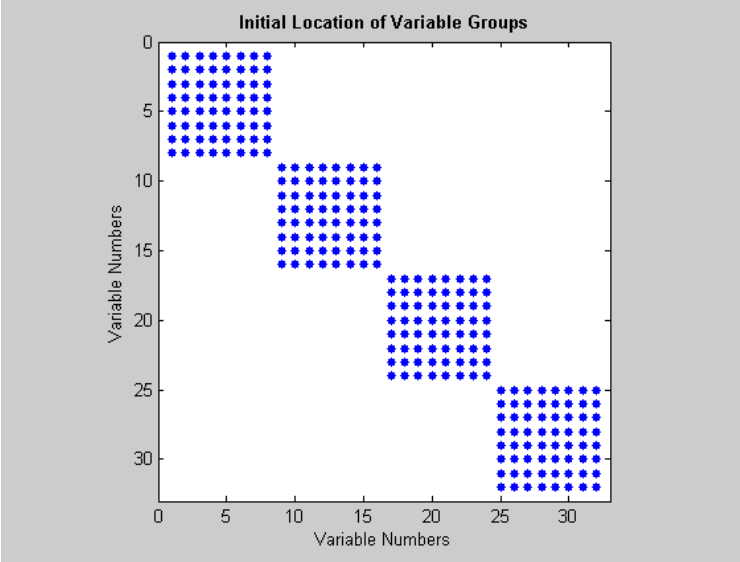
[Groups Ungrouped Removed] = AAGROUP(X) performs the same grouping as described above with CutOff set to 0.7.

Example

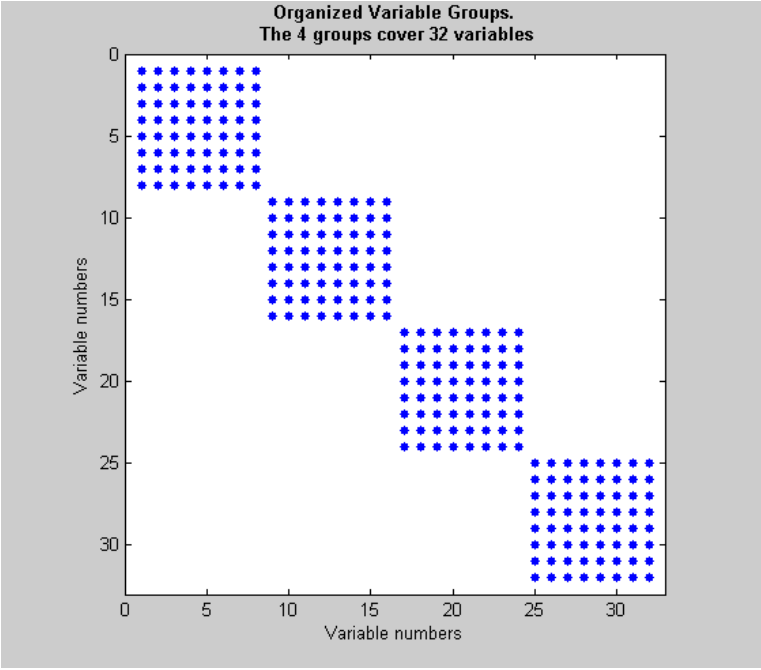
```
clear;

load groupexample;

[groups ungrouped removed] = aagroup(Data)


Variables:36

Observations:1000

Removed Sensors:

33  34  35  36

The number of the removed sensors:4
```

```
groups =

    [1x8 double]    [1x8 double]    [1x8 double]    [1x8 double]

ungrouped =

   Empty matrix: 1-by-0

removed =

    33    34    35    36
```

```
clf reset;
gplot1(groups);
```




```
clf reset;

gplot2(groups);
```



aakr

Autoassociative kernel regression

Syntax

```
XHat = AAKR(XT,X,H)
```

```
XHat = AAKR(XT,X,H,false)
```

Description

This function is a low level implementation of autoassociative kernel regression. This function is intended to be used to bypass the use of model structures or as a template for the development of custom algorithms.

XHat = AAKR(XT,X,H) predicts the X with the standard autoassociative kernel regression algorithm. XT are the training data, X are the query predictors, and H is the kernel bandwidth. XHat are the corrected values for X.

XHat = AAKR(XT,X,H,false) simulates the autoassociative kernel regression model without displaying the wait bar.

The dimensions of XT are MxP, where M is the number of training observations (memory vectors) and P is the number of variables. The dimensions of X and XHat are NxP, where N is the number of query observations.

This functions has been designed for lightweight deployment. This means that this function may be deployed independent of the PEM Toolbox.

Example

```
clear;
```

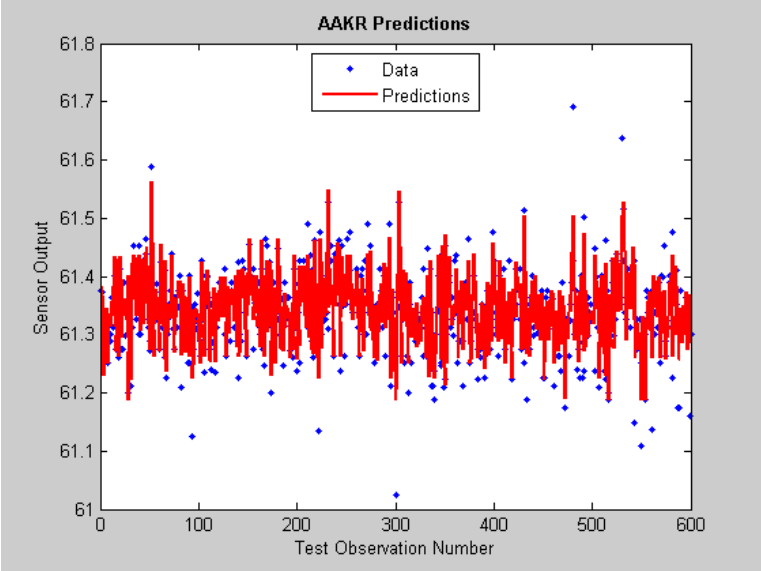
```
load redundantsensors;
```

```
x = cleandata(X(:,[1:4 6:9]));
```

```
train = x(1:200,:);
```

```
test = x(201:800,:);
```

```
p = aakr(train,test,0.5);


clf reset;

plot(test(:,1),'.');

hold on;

plot(p(:,1),'r','LineWidth',2);

hold off;

xlabel('Test Observation Number');

ylabel('Sensor Output');

title('AAKR Predictions','FontWeight','Bold');

legend('Data','Predictions','Location','Best');
```



aars

Autoassociative regression smoothing

Syntax

```
XS = AARS(X,N)
```

```
XS = AARS(Model,X,N)
```

Description

This function smoothes multivariate data using an autoassociative regression model.

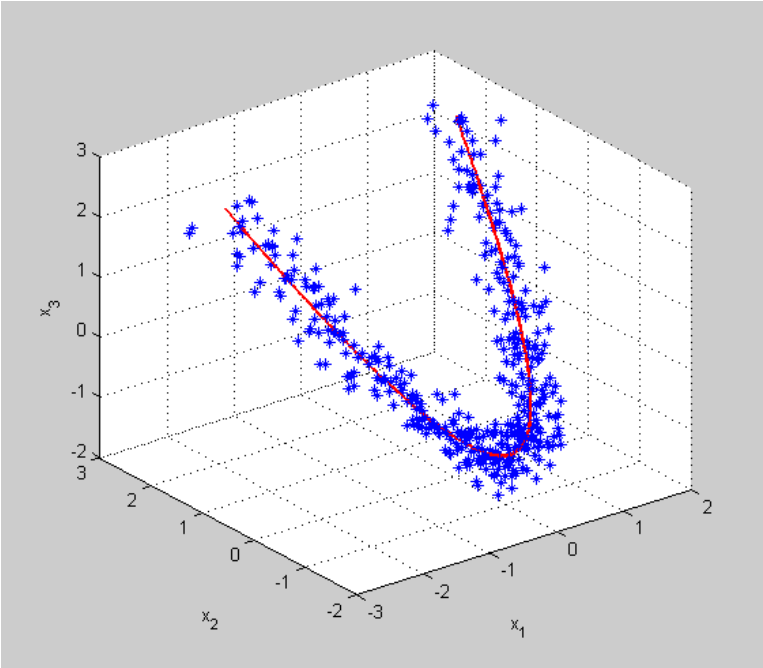
XS = AARS(X,N) smoothes the multivariate data X by successively simulating an autoassociative kernel regression (AAKR) model N times. If N is not supplied, then the default value of 1 is used.

XS = AARS(Model,X,N) smoothes X by successively simulating Model N times. If N is not supplied, then the default value of 1 is used

Example

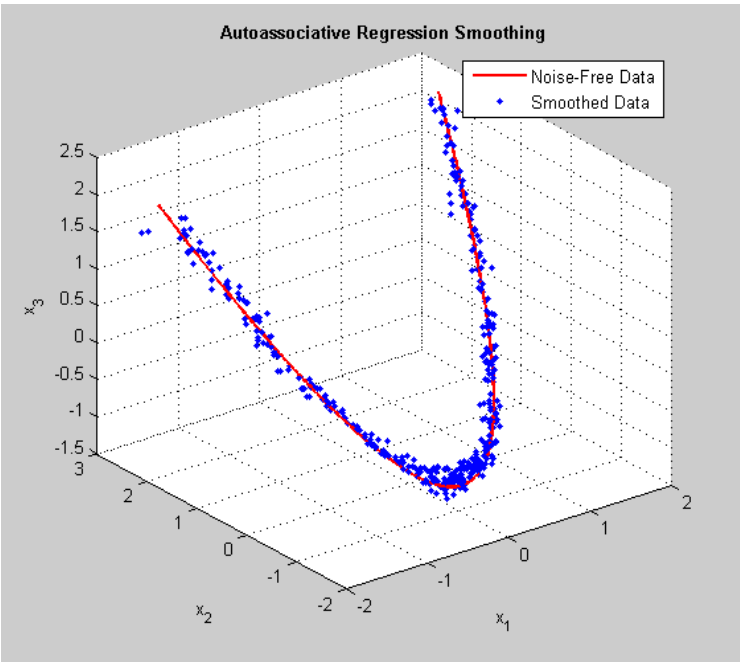
```
clear;

x1 = -20:0.1:20;

x2 = x1.^2-100;

x3 = x1+x2;

xt = [x1' x2' x3'];

[xt xm xstd] = zscore1(xt);

x = xt+0.2.*randn(size(xt));


clf reset;

plot3(x(:,1),x(:,2),x(:,3),'*');

hold on;
```

```
plot3(xt(:,1),xt(:,2),xt(:,3),'r','LineWidth',2);
hold off;
grid on;
xlabel('x_1');
ylabel('x_2');
zlabel('x_3');
```



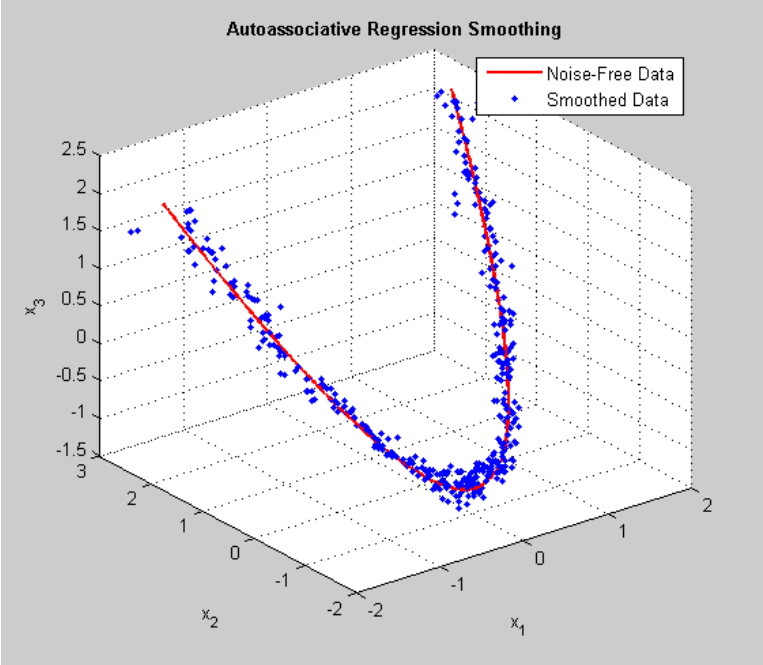

```
xs = aars(x);

clf reset;
plot3(xt(:,1),xt(:,2),xt(:,3),'r','LineWidth',2);
hold on;
plot3(xs(:,1),xs(:,2),xs(:,3),'.');
```

```
hold off;

grid on;

xlim([-2 2]);

ylim([-2 3]);

zlim([-1.5 2.5]);

title('Autoassociative Regression Smoothing','FontWeight','Bold');

xlabel('x_1');

ylabel('x_2');

zlabel('x_3');

legend('Noise-Free Data','Smoothed Data','Location','ne');
```




```
model = initmodel('aakr',x,'vsmethod','a','bandwidth',0.4);

xs = aars(model,x);
```

```
clf reset;
plot3(xt(:,1),xt(:,2),xt(:,3),'r','LineWidth',2);
hold on;
plot3(xs(:,1),xs(:,2),xs(:,3),'.');
hold off;
grid on;
xlim([-2 2]);
ylim([-2 3]);
zlim([-1.5 2.5]);
title('Autoassociative Regression Smoothing','FontWeight','Bold');
xlabel('x_1');
ylabel('x_2');
zlabel('x_3');
legend('Noise-Free Data','Smoothed Data','Location','ne');
```


Autoassociative Regression Smoothing
Noise-Free Data
Smoothed Data
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
x3
3
2
1
0
-1
-2
x2
-2
-1
0
1
2
x1

amfdet

Automated fault detection

Syntax

```
[FHyp FScore Result] = AMFDET(Model,Query)
```

```
[FHyp FScore Result] = AMFDET(Model,Query,'flag',value,...)
```

Description

This function detects faults in a set of query data according to automated methods.

[FHyp FScore Result] = AMFDET(Model,Query) detects faults in Query according to the method and attributes stored in Model.

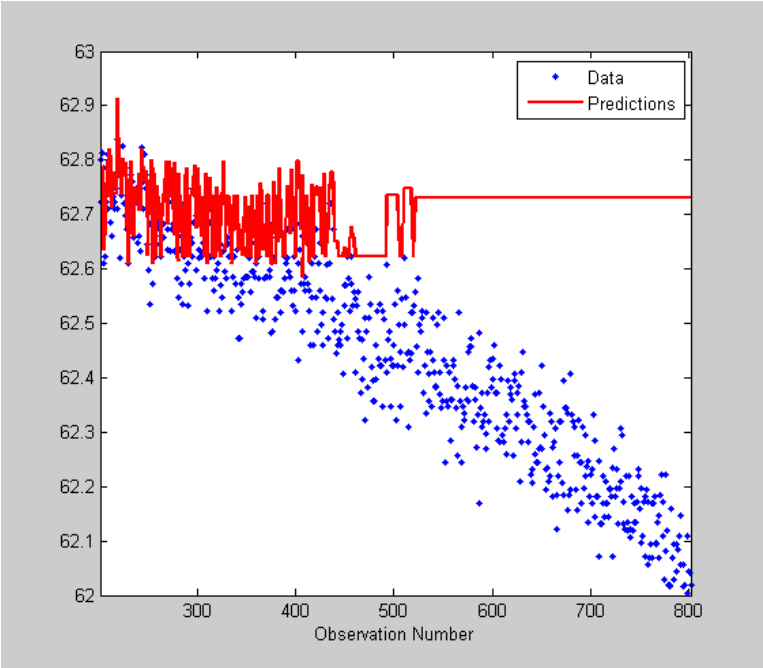
[FHyp FScore Result] = AMFDET(Model,Query,'flag',value,...) detects faults in Query overriding the attributes supplied by the respective flag/value pairs.

This function returns a matrix of fault hypotheses FHyp, fault scores FScore, and results structure Result. Result contains the method dependant information used in the fault detection.

Example

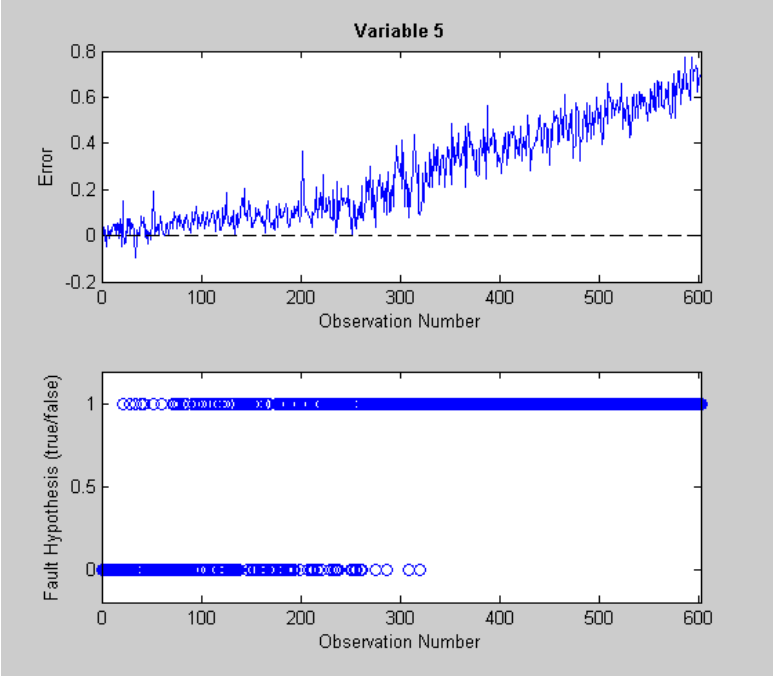
```
clear;

load redundantsensors;

x = cleandata(X);

train = x(1:200,:);

test = x(201:802,:);


model = initmodel('aakr',train);

model = setmsa(model,'plotresults',false,'fdetmethod','sprtn');
```

```
p = runmodel(model,test);


clf reset;

t = 201:802;

plot(t,test(:,5),'b.',t,p(:,5),'r','LineWidth',1.5);

axis([201 802 62 63]);

xlabel('Observation Number');

legend('Data','Predictions');
```




```
model = setmsa(model,'alpha',0.01,'beta',0.10);

model = sprtchar(model);

[fhyp fscore result] = amfdet(model,test);
```

```
clf reset;

sprtplot(result,5);
```



autogroup

Automated variable grouping

Syntax

```
[Groups Ungrouped Removed] = AUTOGROUP(X)

[Groups Ungrouped Removed] = AUTOGROUP(X,CutOff,MinGroupSize)

GData = AUTOGROUP(Data)

GData = AUTOGROUP(Data,CutOff,MinGroupSize)
```

Description

This function uses a set of default settings to perform an automated variable grouping. The grouping methods used in this function include initial grouping (AAGROUP), small group merging (MSGROUP), and orphan variable adoption (AORPHAN).

[Groups Ungrouped Removed] = AUTOGROUP(X) groups the variables of the data matrix X. This function returns a cell array of group indices, where the variable indices of group i are accessed by entering Groups{i}. In addition, the variable indices of the un-grouped and removed variables are returned in Ungrouped and Removed, respectively.

[Groups Ungrouped Removed] = AUTOGROUP(X,CutOff,MinGroupSize) uses the cutoff correlation coefficient CutOff instead of the default 0.7 and uses the minimum group size of MinGroupSize instead of the default values determined by MSGROUP.

GData = AUTOGROUP(Data) groups the variables in the data structure Data and returns a new data structure GData that has its group index values attributes field (groupi) set to the identified groups. The groups include the identified groups as well as groups for the un-grouped and removed variables.

GData = AUTOGROUP(Data,CutOff,MinGroupSize) uses the cutoff correlation coefficient CutOff instead of the default 0.7 and uses the minimum group size of MinGroupSize instead of the default values determined by MSGROUP.

Example

```
clear;
load groupexample;
data = initds(Data,'plotresults',false,'groupeditor',false);
initial_groups = data.attributes.groupi

initial_groups =

    [1x36 double]

data = autogroup(data);
final_groups = data.attributes.groupi

Variables:36
Observations:1000
Removed Sensors:
33  34  35  36
The number of the removed sensors:4

final_groups =

    [1x8  double]      [1x8  double]      [1x8  double]      [1x8  double]
[1x4 double]
```

cleandata

Clean a set of data

Syntax

```
XC = CLEANDATA(X)

XC = CLEANDATA(X,'method')
```

Description

This function cleans a set of data.

XC = CLEANDATA(X) cleans X using the default linear interpolation imputation method ('linear'). This function returns the cleaned data XC.

XC = CLEANDATA(X,'method') cleans X using the supplied imputation method 'method'. 'method' may be set to any of the following character strings.

```
'linear'        linear interpolation

'pca'           principal component analysis
```

If X is a data structure, then XT is also a structure initialized with the attributes of X except for the smoothing method, which is set to the default 'medianfilter'.

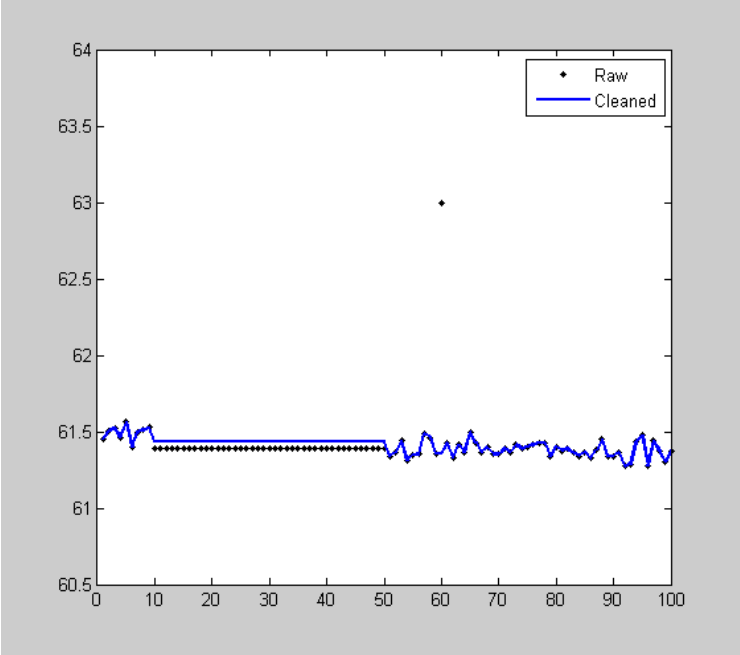
Do not use CLEANDATA on new signals that have not been previously examined to determine if data conditioning is appropriate. The reason being is that some common data contamination mechanisms are significant and therefore should not be corrected.

Also, this function assumes that the data is continuous. If you have highly quantized data it is advised that you smooth the data with local averaging, kernel smoothing (KERS), etc. before presenting it to this function. If you fail to perform this step the quantized data will often be interpreted as being stuck, resulting in the attempted correction of all of the data. This will result in errors. If you do not wish to smooth your data, please use the algorithm's composite functions, specifically PVIMP and UVOUT.

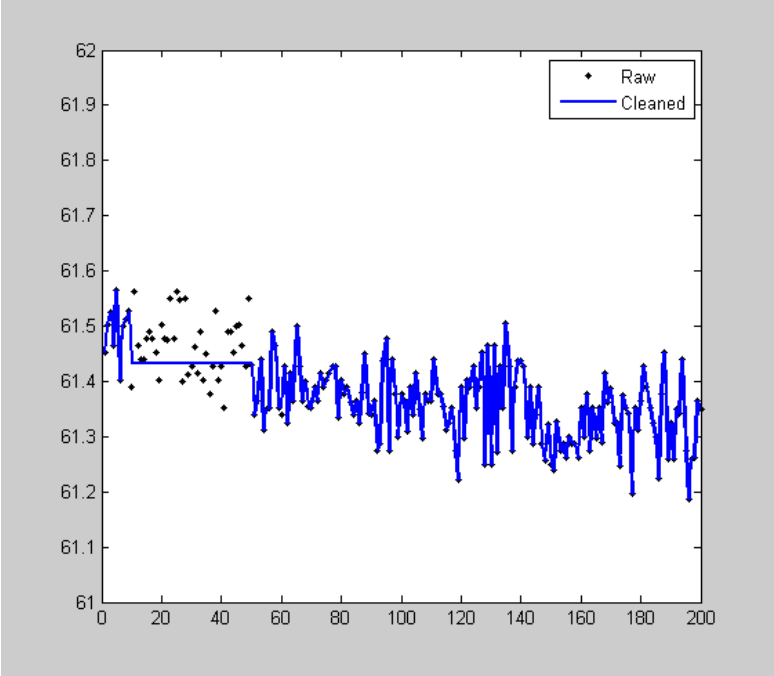
This function detects and corrects NaNs, stuck data, and outliers. The algorithm simply flags each sensor that contains bad data and then uses principal component analysis (PCA) to infer the correct sensor values.

Example

```
clear;

load redundantsensors;

x = X(:,[1:4 6:9]);


x(11:50,1) = x(10,1).*ones(40,1);

x(60,1) = 63;

xc = cleandata(x);

clf reset;

t = 1:100;

plot(t,x(t,1),'k.',t,xc(t,1),'b','LineWidth',1.5);

axis([0 100 60.5 64]);

legend('Raw','Cleaned');
```




```
x = cleandata(X(1:200,[1:4 6:9]));


clf reset;

t = 1:size(x,1);

plot(t,x(t,1),'k.',t,xc(t,1),'b','LineWidth',1.5);

legend('Raw','Cleaned');

axis([0 200 61 62]);
```


Raw
Cleaned
62
61.9
61.8
61.7
61.6
61.5
61.4
61.3
61.2
61.1
61
0
20
40
60
80
100
120
140
160
180
200

confh

Consolidate fault hypotheses

Syntax

```
CFHyp = CONFH(FHyp,Logic)
```

```
CResult = CONFH(Result,Logic)
```

Description

This function consolidates a series of fault hypotheses by removing spurious alarms.

CFHyp = CONFH(FHyp,Logic) consolidates the fault hypotheses FHyp using the supplied alarm logic Logic. If Logic is not supplied then the default of alarm logic of 5 is used.

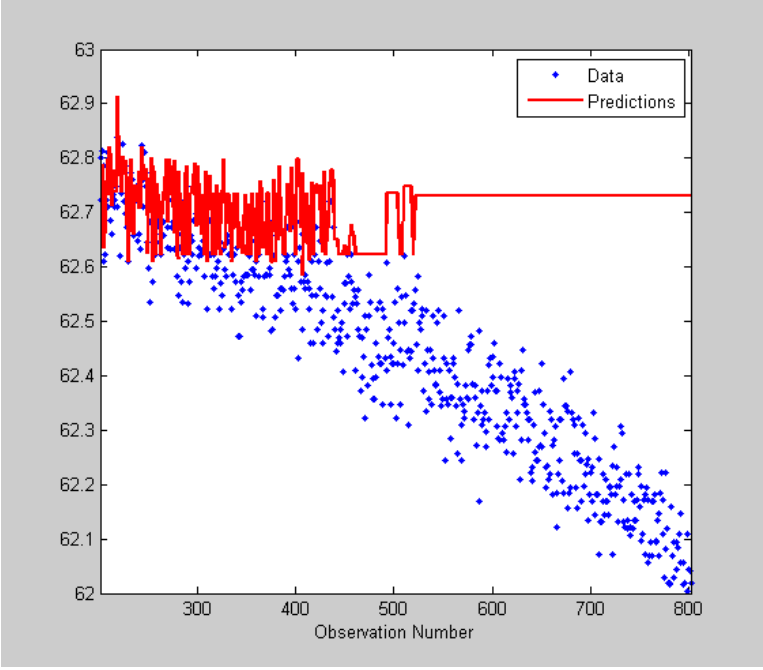
CResult = CONFH(Result,Logic) consolidates the fault hypotheses in Result using the supplied alarm logic Logic. If Logic is not supplied then the default of alarm logic of 5 is used.

Logic may be either a single value that indicates how many successive deviations are used to trigger an alarm or a vector [ND NT], where ND is the number of detections and NT is the total number of observations. For example, a Logic value of 3 would indicate that a signal is faulted when 3 successive observations exceed the threshold and a Logic value of [3 5] would indicate that a signal is faulted when 3 out of 5 observations exceed the threshold.

Example

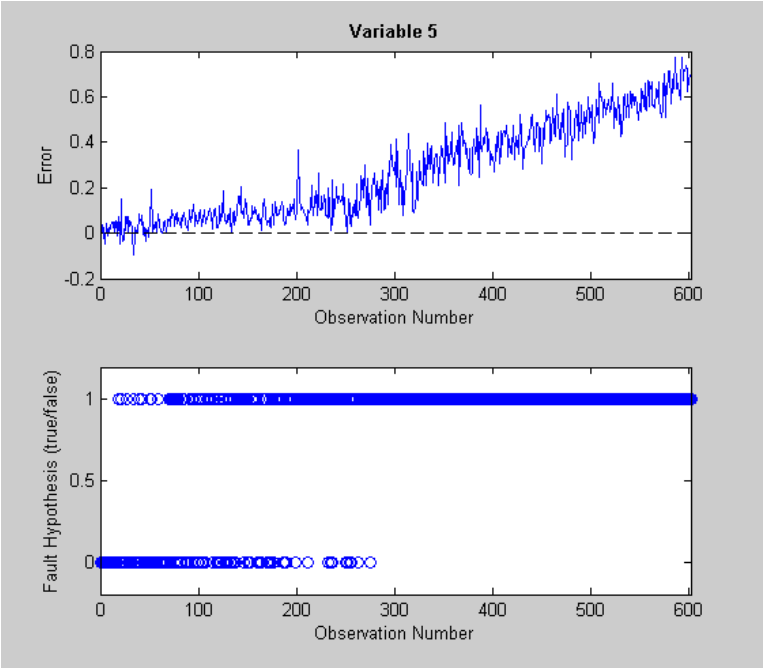
```
clear;

load redundantsensors;

x = cleandata(X);

train = x(1:200,:);

test = x(201:802,:);


model = initmodel('aakr',train);
```

```
model = setmsa(model,'plotresults',false,'fdetmethod','sprtn');


p = runmodel(model,test);


clf reset;

t = 201:802;

plot(t,test(:,5),'b.',t,p(:,5),'r','LineWidth',1.5);

axis([201 802 62 63]);

xlabel('Observation Number');

legend('Data','Predictions');
```

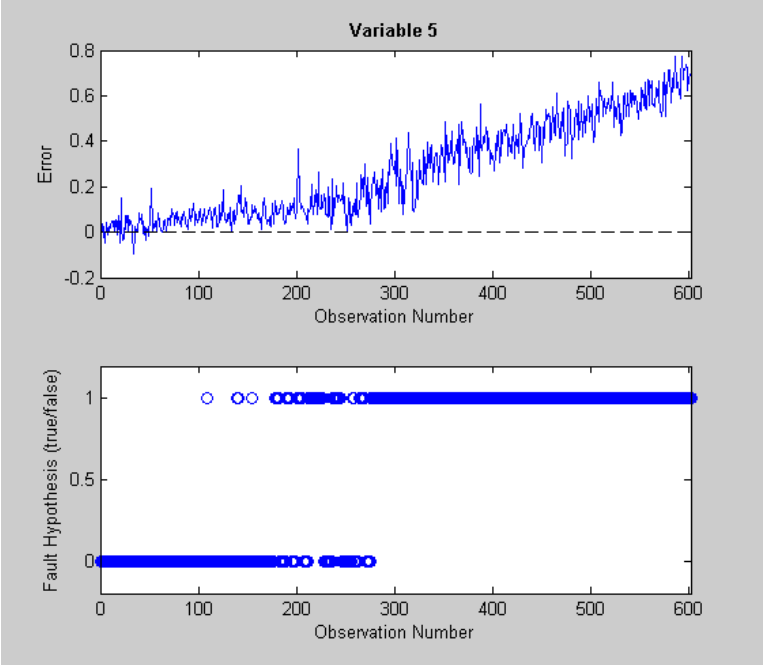



```
model = setmsa(model,'alpha',0.05,'beta',0.10);

model = sprtchar(model);
```

```
[fhyp fscore result] = amfdet(model,test);

sprtplot(result,5);
```




```
result = confh(result);

sprtplot(result,5);
```


Variable 5
Error
0.8
0.6
0.4
0.2
0
-0.2
0
100
200
300
400
500
600
Observation Number
Fault Hypothesis (true/false)
1
0.5
0
0
100
200
300
400
500
600
Observation Number

eulm

EULM fault detection

Syntax

```
[FHyp FScore] = EULM(Error,ErrorU,Tol)
```

Description

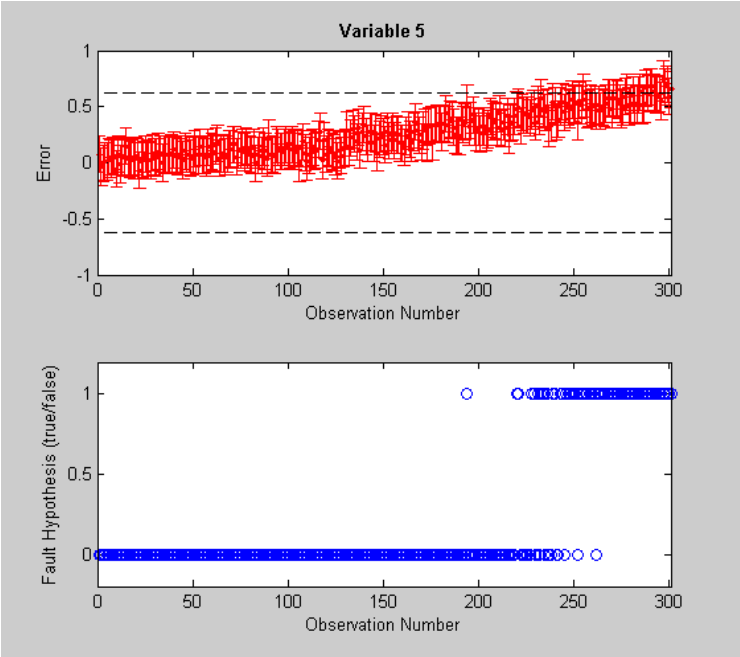
This function performs error uncertainty limit monitoring fault detection.

[FHyp FScore] = EULM(Error,ErrorU,Tol) detects faults by comparing an error Error and its uncertainty ErrorU with a tolerance Tol. Here, Tol is matrix of values that is usually defined as a multiple of the target predictions. For example, if Error is the prediction error of an empirical model for the target data Target, then a 10% tolerance would be equal to 0.1*Target.

Example

```
clear;

load redundantsensors;

x = cleandata(X);

train = x(1:2:200,:);

test = x(201:2:802,:);


model = initmodel('aakr',train);

model = setmsa(model,'plotresults',false,'interval','pi');


[p pu] = modelu(model,train);


p = runmodel(model,test);

pu = ones(size(test,1),1)*mean(pu);
```

```
error = p-test;

erroru = pu;

mean_value = ones(size(test,1),1)*mean(train);

tol = 0.01.*mean_value;


[fhyp fscore] = eulm(error,erroru,tol);


clf reset;

eulmplot(error,erroru,tol,fhyp,5);
```



fdetect

Fault detection

Syntax

```
[FHyp FScore] = FDETECT('method',Parameters...)

[FHyp FScore] = FDETECT('eulm',Error,ErrorU,Target,[Tol])

[FHyp FScore] = FDETECT('sprtn',M,V,Error,[Alpha],[Beta],[Tol])

[FHyp FScore] = FDETECT('uicm',Pred,PredU,Target,[Thresh],[Window])
```

Description

This function uses various methods to detect faults within a set of data.

[FHyp FScore] = FDETECT('method',Parameters...) detects faults by using the specified method 'method' and its associated Parameters. This function returns a matrix of fault hypothesis (true/false) FHyp and their associated scores FScore. The detection method may be set to any of the following character strings:

```
'eulm'      error uncertainty limit monitoring

'sprtn'     SPRT of a normal distribution

'uicm'      uncertainty interval coverage monitoring
```

[FHyp FScore] = FDETECT('eulm',Error,ErrorU,Target,[Tol]) detects faults by comparing the error Error and its uncertainty ErrorU with a specified tolerance Tol. Here, Error is the predictive error of an empirical model for Target and the tolerance Tol is a scalar value that indicates an acceptable fractional range of variation for the predictive error. If not specified the tolerance is set to 0.01 or 1%.

[FHyp FScore] = FDETECT('sprtn',M,V,Error,[Alpha],[Beta],[Tol]) detects faults with the SPRT of a normal distribution using the false alarm probability Alpha, missed alarm probability Beta, and tolerance Tol. Tol may be either a single value or a matrix of values whose length is equal to the number of variables in Error. If not supplied, Alpha is set to 0.01 (1%), Beta is set to 0.10 (10%), and Tol is set to 2 standard deviations of the training error. Refer to sprtn for additional information.

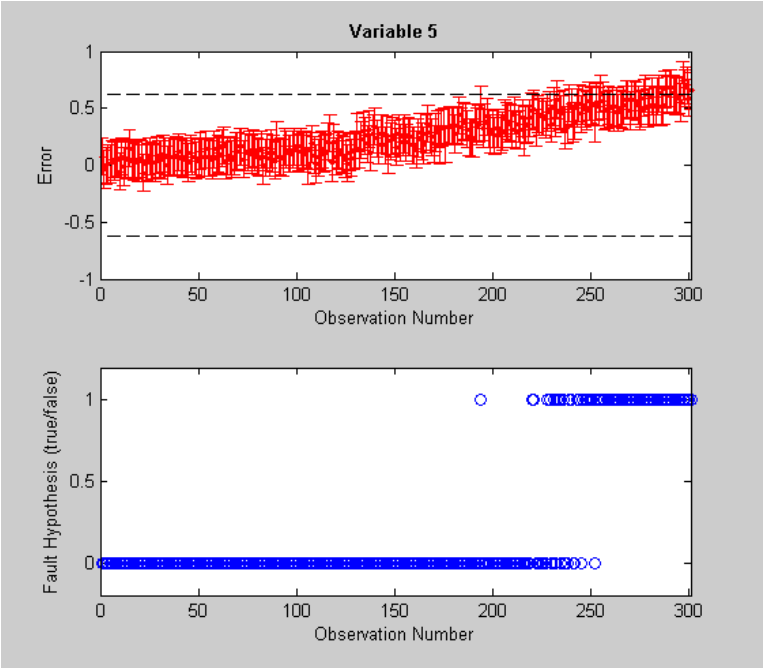
[FHyp FScore] = FDETECT('uicm',Pred,PredU,Target,[Thresh],[Window]) detects faults by comparing the local uncertainty interval, defined by Pred and its uncertainty PredU, coverage of Window observations with a threshold Thresh. If not specified, Thresh is set to 0.5 (50%) and Window is set to 100 for more than 1,000 observations and NumObs/10 otherwise. Here, NumObs is the number of observations in Pred, PredU, and Target. Refer to uicm for additional information.

To use the default value for an optional setting either neglect entering it in the function call or use [ ].

Example

```
clear;

load redundantsensors;

x = cleandata(X);

train = x(1:2:200,:);

test = x(201:2:802,:);


model = initmodel('aakr',train);

model = setmsa(model,'plotresults',false,'interval','pi');


[p pu] = modelu(model,train);


p = runmodel(model,test);

pu = ones(size(test,1),1)*mean(pu);

error = p-test;

erroru = pu;

mean_value = ones(size(test,1),1)*mean(train);
```

```
tol = 0.01.*mean_value;

[fhyp fscore] = fdetect('eulm',error,erroru,test);

clf reset;

eulmplot(error,erroru,tol,fhyp,5);
```



gcplot

Group correlation plot

Syntax

```
GCPLOT(X)

GCPLOT(X,Groups)

GCPLOT(S)
```

Description

This function creates a plot of the absolute correlation coefficients for the organized variable groups.

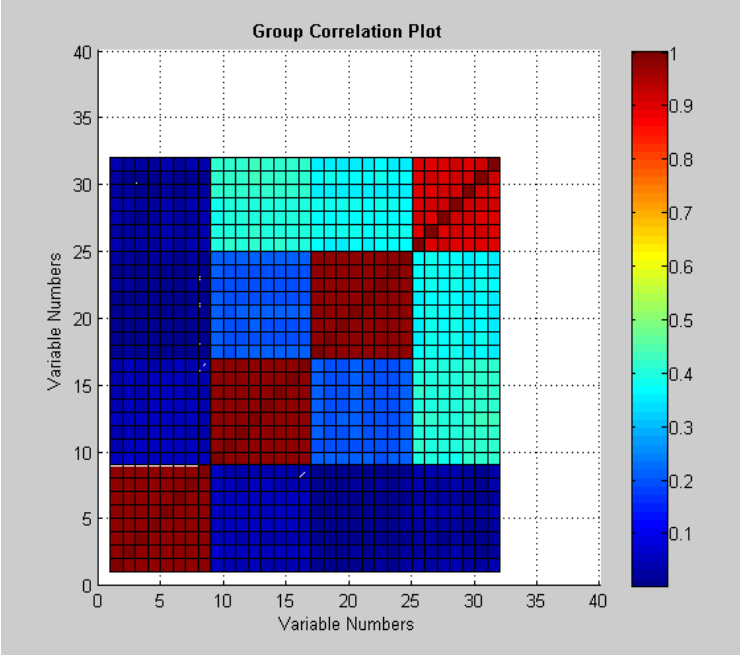
GCPLOT(X) creates a correlation plot for the all variables in X.

GCPLOT(X,Groups) creates a group correlation plot for the organized variable groups of X, where Groups is a cell array of variable indices that define the variable groups.

GCPLOT(S) creates a group correlation plot for the structure S. Here, S may be either a data structure or model structure.

Example

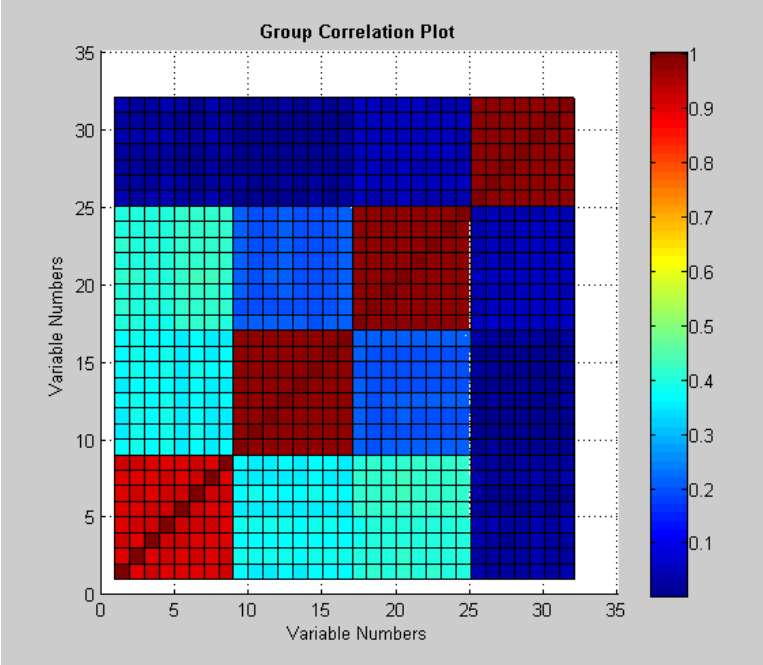
```
clear;

load groupexample;


clf reset;

gcplot(Data);


Warning: Divide by zero.
```




```
[g ug r] = aagroup(Data);
```

```
Variables:36

Observations:1000

Removed Sensors:

33  34  35  36

The number of the removed sensors:4
```

```
clf reset;

gcplot(Data,g);
```


Group Correlation Plot
Variable Numbers
Variable Numbers
0
5
10
15
20
25
30
35
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

initmodel

Initialize model structure

Syntax

```
Model = INITMODEL('aatype',Train)

Model = INITMODEL('aatype',Train,'flag',value,...)

Model = INITMODEL('type',Predictor,Response)

Model = INITMODEL('type',Predictor,Response,'flag',value,...)
```

Description

This function initializes a PEM Toolbox model structure.

Model = INITMODEL('aatype',Train) initializes an autoassociative model of the specified type 'aatype' with the training data Train using the default architecture. The autoassociative model type may be set to any of the following character strings:

```
'aakr'      autoassociative kernel regression

'aann'      autoassociative neural network

'palm'      pseudo-autoassociative linear model
```

Model = INITMODEL('aatype',Train,'flag',value,...) initializes an autoassociative model structure of a specified type from the training data Train and model specific architecture indicated by their appropriate flag/value pairs.

Model = INITMODEL('type',Predictor,Response) initializes an inferential or hetero-associative model of the specified type with the training predictors Predictor and responses Response using the default architecture. The inferential and heteroassociative model type may be set to any of the following character strings:

```
'kr'        kernel regression

'linear'    linear regression
```

Model = INITMODEL('type',Predictor,Response,'flag',value,...) initializes an inferential or hetero-associative model structure of a specified type from the training predictors Predictor, responses Response, and model specific architecture indicated by their appropriate flag/value pairs.

The architecture settings for the supported models may be set by any of the following flag/value pairs, where the default values are contained in parenthesis:

```
Autoassociative Kernel Regression ('aakr')
'bandwidth'     bandwidth (0.5)
'distance'      distance measure ('euclid')
'nmem'          number of memory vectors (1/2 x N)
'vsmethod'      vector selection method ('x')

Autoassociative Neural Network ('aann')
'display'       training display frequency (10)
'nbottleneck'  number of bottleneck neurons (1/2 x P)
'nmap'          number of mapping neurons (P)
'ntrainepochs' number of training epochs (300)
'trainfunction'      neural network training function ('trainlm')

Kernel Regression ('kr')
'bandwidth'     bandwidth (0.5)
'distance'      distance measure ('euclid')
'nmem'          number of memory vectors (1/2 x N)
'vsmethod'      vector selection method ('x')

Linear Regression ('linear')
```

```
'regpar'        regularization parameter (0)


Pseudo-Autoassociative Linear Model ('palm')
'condition'     target condition number (100)
```

Here, N refers to the number of training observations and P refers to the number of variables in the data.

For AAKR and KR, the bandwidth can be either a single value or a vector of P values. For linear regression, the regularization parameter can be either a single value or a vector of P values.

For PEM Toolbox 1R, additional model types are available and may be set by the following character string:

```
'aamset'    autoassociative MSET
'esee'      Expert State Estimation Engine (Expert Microsystems)
'psa'       Parity Space Algorithm (Expert Microsystems)
```

The architecture settings for the alternative model may be set by any of the following flag/value pairs, where the default values are contained in parenthesis:

```
Autoassociative MSET ('aamset')
'bandwidth'     bandwidth (1)
'distance'      distance measure ('euclid')
'nmem'          number of memory vectors (4 x P)
'regpar'        regularization parameter (0)
'vsmethod'      vector selection method ('x')


Expert State Estmation Engine ('esee')
'bandwidth'     bandwidth (0.5)
'nmem'          number of memory vectors (1/2 x N)
```

The distance measure may be set to any of the following character strings:

```
'aeuclid'   adaptive Euclidean
'euclid'    Euclidean
'norm1'     L-1 norm (PEM Toolbox 1R)
'reuclid'      robust Euclidean (PEM Toolbox 1R)
```

The vector selection method may be set to any of the following characters:

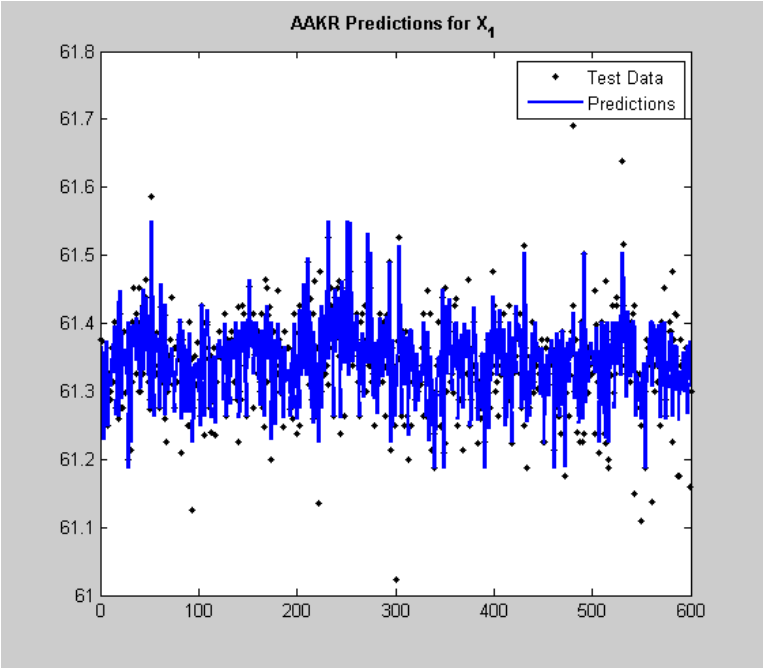
```
'a'         select all
'f'         fuzzy c-means clustering
'h'         Adeli-Hung clustering
'm'         min-max
's'         sort-select
'x'         combination of 'm' and 's'
```

NOTE: The only functions supported for inferential and heteroassociative modeling are INITMODEL and RUNMODEL.

Example

```
clear;
load redundantsensors;
x = cleandata(X(:,[1:4 6:9]));
train = x(1:200,:);
test = x(201:800,:);

model = initmodel('aakr',train);
model = setmsa(model,'plotresults',false);

p = runmodel(model,test);
```

```
clf reset;

t = 1:size(test,1);

plot(t,test(:,1),'k.',t,p(:,1),'b','LineWidth',1.5);

title('AAKR Predictions for X_1','FontWeight','Bold');

legend('Test Data','Predictions');
```



kr

Kernel regression

Syntax

```
YHat = KR(XT,YT,X,H)
```

```
YHat = KR(XT,YT,X,H,false)
```

Description

This function is a low level implementation of kernel regression. This function is intended to be used to bypass the use of model structures or as a template for the development of custom algorithms.

YHat = KR(XT,YT,X,H) predicts the response Y with the standard kernel regression algorithm. XT are the training predictors, YT are the training responses, X are the query predictors, and H is the kernel bandwidth. YHat is the predicted response for the predictors X.

YHat = KR(XT,YT,X,H,false) simulates the kernel regression model without displaying the wait bar.

The dimensions of XT are MxP and YT are MxR, where M is the number of training observations (memory vectors), P is the number of predictor variables and R is the number of response variables. The dimensions of X are NxP and YHat are NxR, where N is the number of query observations.

This functions has been designed for lightweight deployment. This means that this function may be deployed independent of the PEM Toolbox.

Example

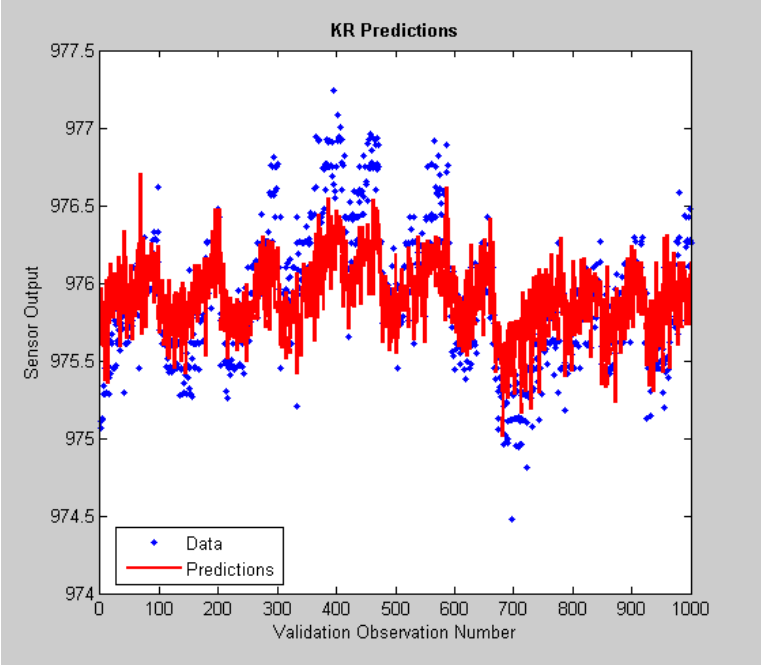
```
clear;
```

```
load nuclearcoolant;
```

```
ptrain = train(:,9:10);
```

```
rtrain = train(:,11:end);
```

```
pval = validation(:,9:10);

rval = validation(:,11:end);


p = kr(ptrain,rtrain,pval,0.5);


clf reset;

plot(rval(:,1),'.');

hold on;

plot(p(:,1),'r','LineWidth',2);

hold off;

xlabel('Validation Observation Number');

ylabel('Sensor Output');

title('KR Predictions','FontWeight','Bold');

legend('Data','Predictions','Location','Best');
```


KR Predictions
977.5
977
976.5
976
975.5
975
974.5
974
Sensor Output
0
100
200
300
400
500
600
700
800
900
1000
Validation Observation Number
Data
Predictions

modchar

Model characterization

Syntax

```
Model = MODCHAR(Model)
```

```
Model = MODCHAR(Model,Query)
```

Description

This function characterizes an empirical model.

Model = MODCHAR(Model) characterizes the empirical model Model by using the data containted in the model structure.

Model = MODCHAR(Model,Query) characterizes Model with Query.

Example

```
clear;
```

```
load redundantsensors;
```

```
x = cleandata(X(:,[1:4 6:9]));
```

```
train = x(1:200,:);
```

```
test = x(201:802,:);
```

```
model = initmodel('aakr',train);
```

```
model = setmsa(model,'plotresults',false,'uncertmethod','analytic');
```

```
attributes = model.attributes
```

```
attributes =
```

```
           accuracy: []
       allowclipping: 1
               alpha: 0.0100
      autosensitivity: []
                beta: 0.1000
                bias: []
          biasmethod: 'mse'
     crosssensitivity: []
             display: 1
       driftvariable: 'all'
               error: []
    eulmdetectability: []
       eulmtolerance: 0.0100
          fdetmethod: 'eulm'
        groupeditor: 1
             groupi: {[1 2 3 4 5 6 7 8]}
         groupnames: {'Group 1'}
        groupoverlap: 0
             interval: 'ci'
    kernelparameters: {[]}
             maxdrift: 2
                 name: 'Model of Data'
       noisevariance:  {[0.0029  0.0028  0.0029  0.0036  0.0038  0.0018
0.0024 0.0028]}
```

```
                nruns: 50
              nsample: 200
          plotresults: 0
               sample: 'train'
        samplemethod: 'direct'
        smoothmethod: 'medianfilter'
           smoothpar: 5
    sprtdetectability: []
          sprtmethod: 'default'
       sprttolerance: 3
       uicmthreshold: 0.5000
          uicmwindow: 20
         uncertainty: []
        uncertmethod: 'analytic'
       variablenames: {1x8 cell}
    waveletparameters: {[]}

model = modchar(model,test);
attributes = model.attributes

attributes =

            accuracy: [1x8 double]
       allowclipping: 1
```

```
                alpha: 0.0100

      autosensitivity:  [0.4066  0.4933  0.3918  0.6354  0.5241  0.5263
0.5254 0.5624]

                 beta: 0.1000

                 bias: [0 0 0 0 0 0 0 0]

           biasmethod: 'mse'

     crosssensitivity:  [0.2200  0.2513  0.2306  0.2539  0.2003  0.2643
0.2248 0.2470]

              display: 1

        driftvariable: 'all'

                error: [1x1 struct]

    eulmdetectability:  [0.0030  0.0034  0.0029  0.0052  0.0042  0.0029
0.0033 0.0039]

        eulmtolerance: 0.0100

           fdetmethod: 'eulm'

          groupeditor: 1

               groupi: {[1 2 3 4 5 6 7 8]}

           groupnames: {'Group 1'}

         groupoverlap: 0

             interval: 'ci'

     kernelparameters: {[]}

             maxdrift: 2

                 name: 'Model of Data'

        noisevariance:  {[0.0029  0.0028  0.0029  0.0036  0.0038  0.0018
0.0024 0.0028]}

                nruns: 50
```

```
              nsample: 200
           plotresults: 0
                sample: 'train'
         samplemethod: 'direct'
         smoothmethod: 'medianfilter'
            smoothpar: 5
    sprtdetectability:  [0.0037  0.0036  0.0029  0.0041  0.0035  0.0027
0.0032 0.0034]
           sprtmethod: 'default'
        sprttolerance: 2.4421
        uicmthreshold: 0.5000
           uicmwindow: 20
          uncertainty:  [0.1080  0.1060  0.1078  0.1207  0.1226  0.0845
0.0976 0.1057]
         uncertmethod: 'analytic'
        variablenames: {1x8 cell}
    waveletparameters: {[]}
```

optmodel

Optimize model architecture

Syntax

```
BestModel = OPTMODEL(Model,Query,'method','flag',values,...)
```

Description

This function optimizes the architecture of an empirical model.

BestModel = OPTMODEL(Model,Query,'method','flag',values,...) optimizes Model with Query according to the specified method and model specific architecture settings defined by the flag/value pairs (see initmodel). The optimization method may be set by any of the following character strings:

```
'error'         error
'uncertainty'   uncertainty
```

For architecture settings that are strings, the values must be entered within {}.

This function performs a combinatorial grid search of the provided architecture settings and returns the model that has the minimum error or uncertainty.

Example

```
clear;

load redundantsensors;

x = cleandata(X(:,[1:4 6:9]));

train = x(1:200,:);

test = x(201:800,:);


model = initmodel('aakr',train);

model = setmsa(model,'plotresults',false);
```

```
starting_architecture = model.architecture


starting_architecture =


    bandwidth: 0.5000
     distance: 'euclid'
         nmem: 100
         type: 'aakr'
     vsmethod: 'x'



p = runmodel(model,test);
starting_error = mean(mean((p-test).^2))


starting_error =


    0.0017


h = [0.2 0.5 1.0 2.0];
n = [25 50 100];
v = {'x' 'm' 's'};


bestmodel                                                                =
optmodel(model,test,'error','bandwidth',h,'nmem',n,'vsmethod',v);
```

```
best_architecture = bestmodel.architecture
```

```
best_architecture =
```

```
    bandwidth: 1
     distance: 'euclid'
         nmem: 100
         type: 'aakr'
     vsmethod: 'm'
```

```
p = runmodel(bestmodel,test);
best_error = mean(mean((p-test).^2))
```

```
best_error =
```

```
    0.0010
```

```
percent_improvement = ((starting_error-best_error)/starting_error)*100
```

```
percent_improvement =
```

```
   38.7391
```

runmodel

Run model simulation

Syntax

```
[P R] = RUNMODEL(Model,Query)

[P R] = RUNMODEL(Model,Query,Display,PlotResults)

[P R] = RUNMODEL(Model,Predictor)

[P R] = RUNMODEL(Model,Predictor,Response)

[P R] = RUNMODEL(Model,Predictor,Response,Display,PlotResults)
```

Description

This function runs a simulation of an empirical model for a set of query data.

[P R] = RUNMODEL(Model,Query) simulates the autoassociative model Model with the query data Query. This function returns the model predictions P and a reliability metric R. The reliability metric R can have values on [0,1], where a value of 1 indicates a reliable estimate.

[P R] = RUNMODEL(Model,Query,Display,PlotResults) runs simulation of an autoassociative model and overrides the display and plotresults attributes of Model.

[P R] = RUNMODEL(Model,Predictor) simulates the inferential model Model with the query predictor variables Predictor. This usage does not plot the results of the simulation since the target values are not supplied.

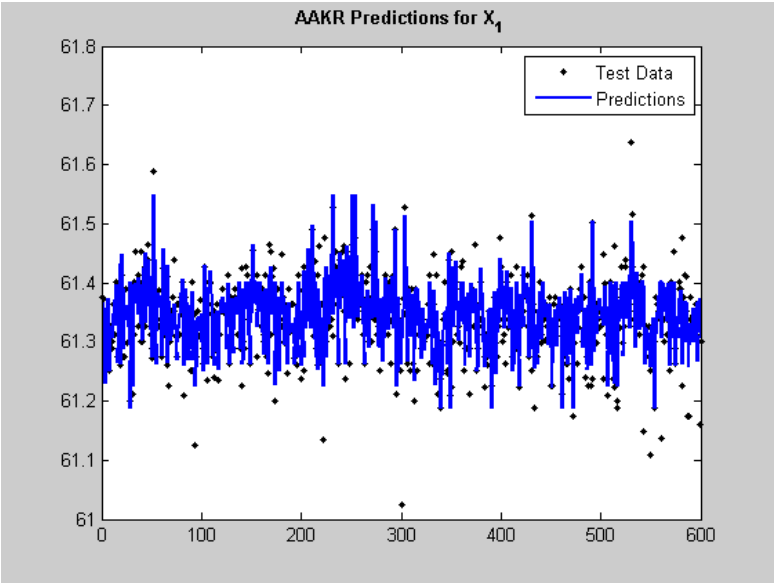
[P R] = RUNMODEL(Model,Predictor,Response) simulates the inferential model Model with the query predictor variables Predictor and their target responses Response.

[P R] = RUNMODEL(Model,Predictor,Response,Display,PlotResults) runs simulation of an inferential model and overrides the display and plotresults attributes of Model.

Example

```
clear;

load redundantsensors;
```

```
x = cleandata(X(:,[1:4 6:9]));

train = x(1:200,:);

test = x(201:800,:);


model = initmodel('aakr',train);

model = setmsa(model,'plotresults',false);


p = runmodel(model,test);


clf reset;

t = 1:size(test,1);

plot(t,test(:,1),'k.',t,p(:,1),'b','LineWidth',1.5);

title('AAKR Predictions for X_1','FontWeight','Bold');

legend('Test Data','Predictions');
```



setmsa

Set model structure attributes

Syntax

```
Model = SETMSA(Model,'attribute',value,...)
```

Description

This function sets one or more of a model's attributes.

Model = SETMSA(Model,'attribute',value,...) sets a series of model structure attributes to the supplied values.

The model structure attributes may be set by any of the following flag/value pairs, where the default values are contained in parenthesis:

```
'allowclipping'     allow prediction clipping (true)
'alpha'             SPRT false alarm probability (0.01)
'beta'              SPRT missed alarm probability (0.1)
'biasmethod'        bias estimation method ('mse')
'display'           display progress (true)
'driftvariable'     model metric drift variables ('all')
'eulmtolerance'     EULM tolerance (0.01)
'fdetmethod'        fault detection method ('eulm')
'groupeditor'       launch group editor after grouping (true)
'groupi'            group indices ([1 2 ... NumVar])
'groupnames'        group names ('Group 1')
'groupoverlap'      allow group overlapping (false)
'interval'          uncertainty interval ('ci')
'maxdrift'          model metric maximum drift in STD (2)
```

```
   'name'              data set name ('Data')
   'nruns'             number of iterations (50)
   'nsample'           number of samples (NumTrainObs)
   'plotresults'       plot results (true)
   'sample'            Monte Carlo sample definition ('train')
   'samplemethod'      noise sampling method ('direct')
   'smoothmethod'      smoothing method ('medianfilter')
   'smoothpar'         smoothing parameter (5)
   'sprtmethod'        SPRT detection method ('default')
   'sprttolerance'     SPRT mean tolerance (3)
   'uicmthreshold'     UICM threshold (0.50)
   'uicmwindow'        UICM window (100 if NumObs>1,000 or NumObs/10)
   'uncertmethod'      uncertainty estimation method ('analytic')
   'variablenames'     variable names ('Variable i' & i = 1-NumVar)
```

Example

```
clear;
load redundantsensors;
model = initmodel('aakr',X(1:200,[1:4 6:9]));
model = newgroup(model,1:3,'First Group');
model = newgroup(model,4:6,'Second Group');
model = newgroup(model,7:8,'Third Group');
attributes = model.attributes


attributes =
```

```
            accuracy: []
        allowclipping: 1
                alpha: 0.0100
      autosensitivity: []
                 beta: 0.1000
                 bias: []
           biasmethod: 'mse'
     crosssensitivity: []
              display: 1
        driftvariable: 'all'
                error: []
    eulmdetectability: []
        eulmtolerance: 0.0100
           fdetmethod: 'eulm'
         groupeditor: 1
              groupi: {[1 2 3]  [4 5 6]  [7 8]}
          groupnames: {'First Group'  'Second Group'  'Third Group'}
         groupoverlap: 0
             interval: 'ci'
    kernelparameters: {[]}
             maxdrift: 2
                 name: 'Model of Data'
        noisevariance: {[0.0142 0.0131 0.0134]  [0.0036 0.0038 0.0026]
[0.0026 0.0028]}
```

```
                nruns: 50
              nsample: 200
          plotresults: 1
               sample: 'train'
         samplemethod: 'direct'
         smoothmethod: 'medianfilter'
            smoothpar: 5
    sprtdetectability: []
           sprtmethod: 'default'
        sprttolerance: 3
        uicmthreshold: 0.5000
           uicmwindow: 20
          uncertainty: []
         uncertmethod: 'analytic'
        variablenames: {1x8 cell}
    waveletparameters: {[]}

model = setmsa(model,'smoothmethod','kernel','smoothpar',2);
model = setmsa(model,'interval','pi','sample','both');
attributes = model.attributes

attributes =

            accuracy: []
```

```
        allowclipping: 1
                alpha: 0.0100
       autosensitivity: []
                 beta: 0.1000
                 bias: []
           biasmethod: 'mse'
      crosssensitivity: []
              display: 1
        driftvariable: 'all'
                error: []
     eulmdetectability: []
        eulmtolerance: 0.0100
           fdetmethod: 'eulm'
         groupeditor: 1
              groupi: {[1 2 3]  [4 5 6]  [7 8]}
          groupnames: {'First Group'  'Second Group'  'Third Group'}
         groupoverlap: 0
             interval: 'pi'
      kernelparameters: {[1x1 struct]  [1x1 struct]  [1x1 struct]}
             maxdrift: 2
                 name: 'Model of Data'
        noisevariance: {[0.0161 0.0155 0.0154]  [0.0085 0.0072 0.0042]
[0.0042 0.0044]}
                nruns: 50
```

```
           nsample: 200
        plotresults: 1
             sample: 'both'
       samplemethod: 'direct'
       smoothmethod: 'kernel'
          smoothpar: 2
 sprtdetectability: []
         sprtmethod: 'default'
     sprttolerance: 3
     uicmthreshold: 0.5000
        uicmwindow: 20
       uncertainty: []
      uncertmethod: 'analytic'
     variablenames: {1x8 cell}
 waveletparameters: {[]}
```

sprtn

SPRT fault detection for a multivariate normal distribution

Syntax

```
[FHyp FScore] = SPRTN(M,V,Error)

[FHyp FScore] = SPRTN(M,V,Error,Alpha,Beta,Tol)

[FHyp FScore] = SPRTN(M,V,Error,'method')

[FHyp FScore] = SPRTN(M,V,Error,'method',Alpha,Beta,Tol)
```

Description

This function uses the Sequential Probability Ratio Test (SPRT) for a multivariate normal distribution to detect anomalies.

[FHyp FScore] = SPRTN(M,V,Error) detects anomalies with the SPRT by comparing the mean of the multivariate error distribution Error to a training distribution with mean M and variance V. The default false alarm probability (Alpha) of 0.01 (1%), missed alarm probabilitiy (Beta) of 0.1 (10%), and mean tolerance of 3 standard deviations are used.

[FHyp FScore] = SPRTN(M,V,Error,Alpha,Beta,Tol) detects anomalies with the SPRT using the supplied false alarm probability Alpha, missed alarm probability Beta, and mean tolerance Tol in standard deviations (STD) (i.e. a tolerance of 2 corresponds to 2 STDs). If any of the options are not supplied their respective default value will be used.

[FHyp FScore] = SPRTN(M,V,Error,'method') detects anomalies according to the specified method. The detection method may be set to any of the following character strings:

```
'chien'     Chien's one-sided test

'default'   default two-sided test
```

[FHyp FScore] = SPRTN(M,V,Error,'method',Alpha,Beta,Tol) detects anomalies according to the specified method using the supplied detection options.

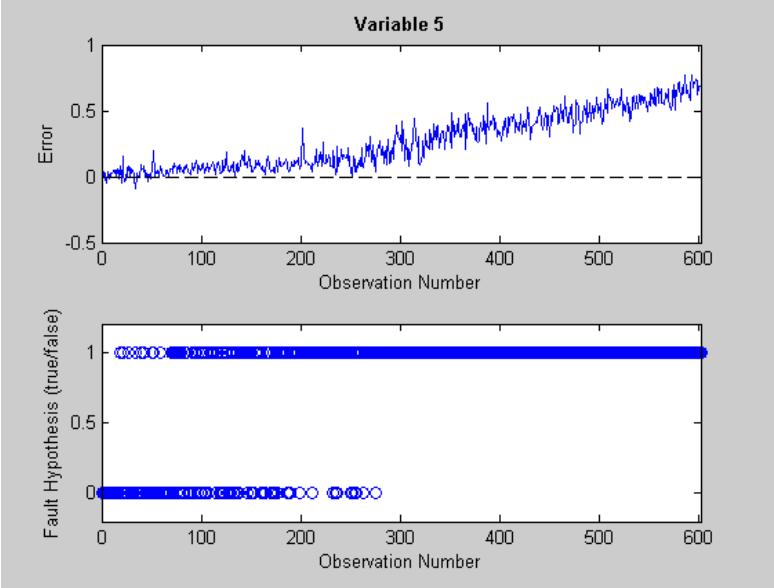
This function returns a matrix of fault hypotheses FHyp and a matrix of fault scores FScores which identify the character of the fault. The values in FScores are the sum of the indicators for the four tests performed by the SPRT. The indicators may be any of the following integer values.

```
positive shift in mean (+M)        1
negative shift in mean (-M)       10
```

Example

```
clear;

load redundantsensors;

x = cleandata(X);

train = x(1:200,:);

test = x(201:802,:);


model = initmodel('aakr',train);

model = setmsa(model,'plotresults',false);


dev = usmoddata(model,model.data.sdev);

deverror = runmodel(model,dev)-dev;


alpha = 0.05;

beta = 0.10;

[m v t] = trainsprt(deverror,deverror,alpha,beta);


testerror = runmodel(model,test)-test;

[fhyp fscore] = sprtn(m,v,testerror,alpha,beta,t);
```

```
clf reset;

sprtplot(m,testerror,fhyp,5)
```



uicm

UICM fault detection

Syntax

```
[FHyp FScore] = UICM(Pred,PredU,Target,Thresh,Window)
```

Description

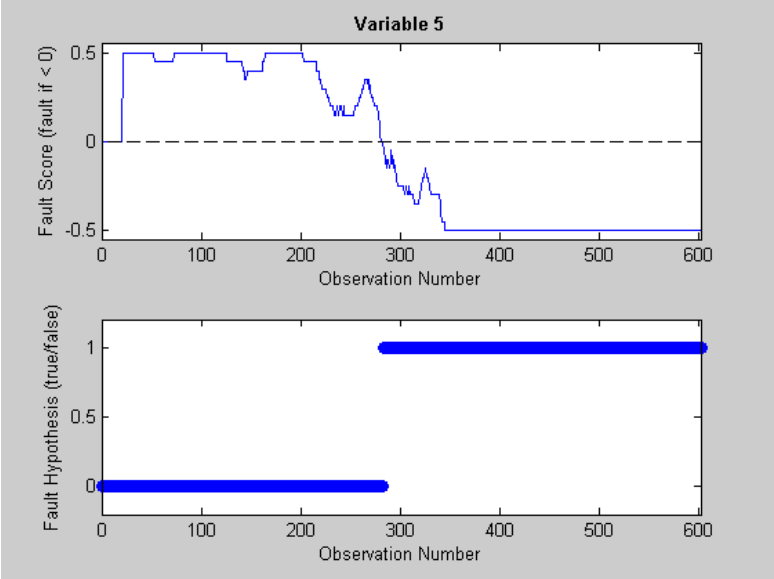
This function performs uncertainty interval coverage monitoring fault detection.

[FHyp FScore] = UICM(Pred,PredU,Target,Thresh,Window) detects faults by comparing the local uncertainty interval, defined by Pred and its uncertainty PredU, coverage of Window observations with a threshold Thresh.

Example

```
clear;

load redundantsensors;

x = cleandata(X);

train = x(1:200,:);

test = x(201:802,:);


model = initmodel('aakr',train);

model = setmsa(model,'plotresults',false,'interval','pi');


p = runmodel(model,test);

model = modchar(model);

pu = ones(size(test,1),1)*model.attributes.uncertainty;

[fhyp fscore] = uicm(p,pu,test,0.5,20);
```

```
clf reset;

uicmplot(fhyp,fscore,5);
```



unscore

Un-scale data

Syntax

```
X = UNSCORE(XS,XMean,XSTD)
```

Description

This function un-scales mean center, unit variance scaled data.

X = UNSCORE(XS,XMean,XSTD) un-scales XS with the scaling means XMean and standard deviations XSTD.

The dimension of X and XS is NumObs x NumVars, where NumObs is the number of observations and NumVars is the number of variables. The dimension of XMean and XSTD is 1 x NumVar.

Example

```
clear;

load redundantsensors;

[xs xm xstd] = zscore1(X);


m = mean(xs)




m =


  1.0e-011 *
```

```
    0.0090   -0.0466   -0.1305    0.1928   -0.0167    0.0522   -0.0341
0.1083   -0.0587
```

```
v = var(xs)
```

```
v =
```

```
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
1.0000    1.0000
```

```
x = unscore(xs,xm,xstd);
```

```
m = mean(x)
```

```
m =
```

```
   61.3521   62.1441   61.8044   63.4647   62.5148   61.8944   61.0898
61.6286   61.9673
```

```
v = var(x)
```

```
v =

    0.0145    0.0137    0.0137    0.0055    0.0497    0.0043    0.0041
0.0047    0.0035
```

vectsel

Vector selection

Syntax

```
[S SI NSI] = VECTSEL(X,'method',N)
```

Description

This function selects representative vectors from a set of data.

[S SI NSI] = VECTSEL(X,'method',N) selects the N representative vectors from X according to 'method'. This function returns the selected vectors S, the indices of the selected vectors SI, and the indices that were not selected NSI.

The dimension of X is MxP, where M is the number of observations and P is the number of variables. The dimension of S is NxP, SI is 1xN, and NSI is 1x(M-N).

The selection method may be set to any of the following characters:

'a'	select all
'f'	fuzzy c-means clustering
'h'	Adeli-Hung clustering
'm'	min-max
's'	sort-select
'x'	combination of 'm' and 's'

For fuzzy c-means clustering the number of memory vectors controls the number of cluster centers. The number of selected vectors may or may not be equal to N.

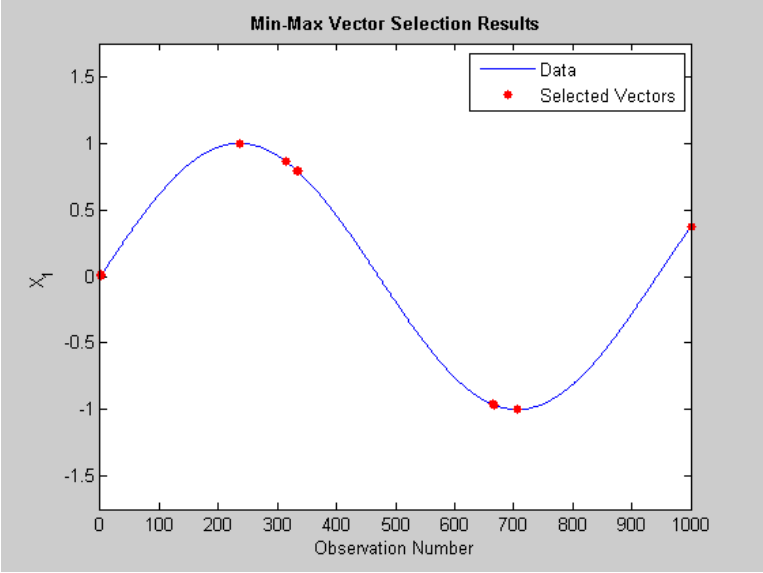
Example

```
clear;

t = 1:1:1000;
```

```
x1 = sin(t./150);

x2 = 1.2.*sin(t./200);

x = [x1' x2'];


[sv si] = vectsel(x,'m',10)



sv =


    0.0067    0.0060

    1.0000    1.1095

    0.0133    0.0120

    0.8666    1.2000

   -0.9631   -0.2248

    0.7925    1.1941

   -0.9613   -0.2189

    0.7884    1.1935

   -1.0000   -0.4600

    0.3742   -1.1507



si =


  Columns 1 through 8
```

```
           1           236           2           314           666
334          665          335

  Columns 9 through 10

         707        1000
```

```
clf reset;
plot(t,x(:,1));
hold on;
plot(si,sv(:,1),'r.','MarkerSize',15);
hold off;
axis([0 1000 -1.75 1.75]);
xlabel('Observation Number');
ylabel('X_1');
title('Min-Max Vector Selection Results','FontWeight','Bold');
legend('Data','Selected Vectors');
```

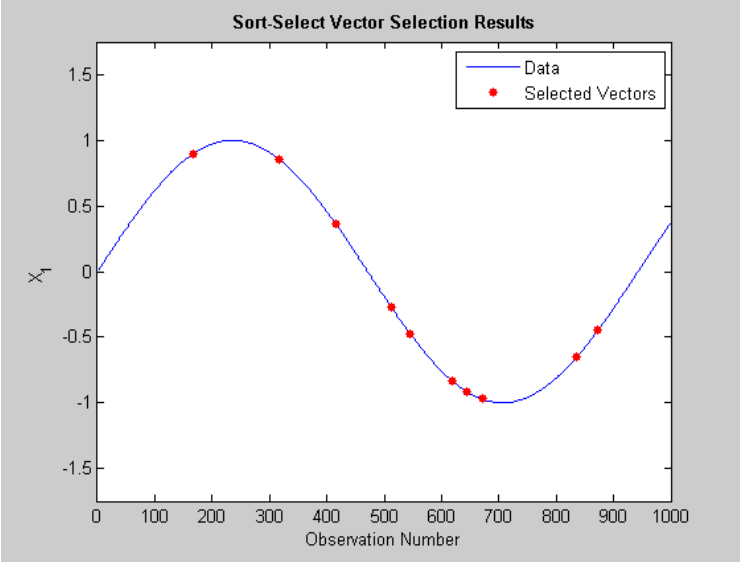



```
[sv si] = vectsel(x,'s',10)
```

```
sv =

   -0.4780    0.4801
   -0.2684    0.6592
   -0.8296    0.0619
   -0.9135   -0.0940
   -0.9716   -0.2541
    0.3662    1.0507
   -0.4468   -1.1283
   -0.6517   -1.0339
    0.9001    0.8936
    0.8564    1.1999
```

```
si =

   546   512   618   644   671   415   873   836   168   317
```

```
clf reset;
plot(t,x(:,1));
hold on;
plot(si,sv(:,1),'r.','MarkerSize',15);
hold off;
axis([0 1000 -1.75 1.75]);
xlabel('Observation Number');
ylabel('X_1');
title('Sort-Select Vector Selection Results','FontWeight','Bold');
legend('Data','Selected Vectors');
```

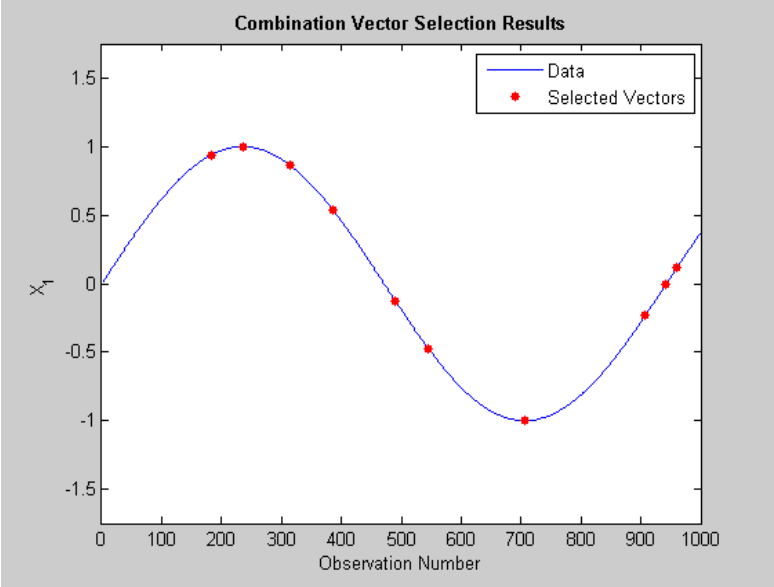



```
[sv si] = vectsel(x,'x',10)
```

```
sv =

   -1.0000   -0.4600
    1.0000    1.1095
   -0.0032   -1.2000
    0.8666    1.2000
   -0.4780    0.4801
   -0.1247    0.7653
    0.1165   -1.1954
   -0.2343   -1.1812
    0.5325    1.1213
    0.9414    0.9547
```

```
si =

   707   236   942   314   546   490   960   907   387   184
```

```
clf reset;
plot(t,x(:,1));
hold on;
plot(si,sv(:,1),'r.','MarkerSize',15);
hold off;
axis([0 1000 -1.75 1.75]);
xlabel('Observation Number');
ylabel('X_1');
title('Combination Vector Selection Results','FontWeight','Bold');
legend('Data','Selected Vectors');
```


Combination Vector Selection Results
Data
Selected Vectors
1.5
1
0.5
0
-0.5
-1
-1.5
x_1
0
100
200
300
400
500
600
700
800
900
1000
Observation Number

vensample

Venetian blind sampling

Syntax

```
[X1 X2] = VENSAMPLE(X)

[X1 X2 ... XN] = VENSAMPLE(X,N)

[X1 X2 ... XN SI] = VENSAMPLE(X,N)
```

Description

This function performs a Venetian blind sampling of a set of data.

[X1 X2] = VENSAMPLE(X) samples X by breaking it into 2 consecutive bands. X1 contains the first half of X and X2 contains the second half of X.

[X1 X2 ... XN] = VENSAMPLE(X,N) samples X by breaking N consecutive bands. Here, [X1 X2 ... XN] are the N samples of X. The number of observations in each band is NumObs/N.

[X1 X2 ... XN SI] = VENSAMPLE(X,N) samples X returning the samples and the indices of the samples in a cell array SI, where SI{i} contains the indices of the ith sample.

If NumObs is not a multiple of N, then the remaining observations are added to the last sample. For example, if you have 101 observations and would like 10 samples (i.e. N = 10), then the first nine samples would contain 10 observations and the tenth sample would conatin 11 observations.

X may be either a matrix or data structure. If X is a data structure, then [X1 X2 ... XN] are similar structures.

Example

```
clear;

t = 1:1:1000;

x1 = sin(t./150);

x2 = 1.2.*sin(t./200);
```

```
xtarget = [x1' x2'];
x = xtarget+0.2.*randn(size(xtarget));

[y1 y2 y3 si] = vensample(x,3);
sample_indices = si



sample_indices =

    [333x1 double]    [333x1 double]    [334x1 double]




data = initds(x);
[data_half1 data_half2] = vensample(data)



data_half1 =

       attributes: [1x1 struct]
             mean: [0.5993 0.8689]
            sdata: [500x2 double]
              std: [0.3944 0.3798]
    structuretype: 'data'
```

```
data_half2 =

       attributes: [1x1 struct]
             mean: [-0.5765 -0.5040]
            sdata: [500x2 double]
              std: [0.4354 0.6532]
    structuretype: 'data'
```

zscore1

Scale data

Syntax

```
[XS XMean XSTD] = ZSCORE1(X)

XS = ZSCORE1(X,XMean,XSTD)
```

Description

This function mean center, unit variance (MCUV) scales data.

[XS XMean XSTD] = ZSCORE1(X) MCUV scales X, returning the scaled data XS, scaling mean XMean, and scaling standard deviation XSTD.

XS = ZSCORE1(X,XMean,XSTD) MCUV scales X with the scaling mean XMean and scaling standard deviation XSTD. This implementation is used to scale new data according to traininig parameters.

The dimension of X and XS is NumObs x NumVars, where NumObs is the number of observations and NumVars is the number of variables. The dimension of XMean and XSTD is 1 x NumVar.

Example

```
clear;

load redundantsensors;

[xs xm xstd] = zscore1(X);


m = mean(xs)



m =
```

```
  1.0e-011 *

    0.0090   -0.0466   -0.1305    0.1928   -0.0167    0.0522   -0.0341
0.1083   -0.0587




v = var(xs)



v =

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
1.0000    1.0000



x = unscore(xs,xm,xstd);

m = mean(x)



m =
```

```
   61.3521    62.1441    61.8044    63.4647    62.5148    61.8944    61.0898
61.6286    61.9673
```

```
v = var(x)
```

```
v =
```

```
    0.0145    0.0137    0.0137    0.0055    0.0497    0.0043    0.0041
0.0047    0.0035
```

PEP Functions List

The Process and Equipment Prognostics (PEP) Toolbox is designed to work with the previously developed Process and Equipment Monitoring (PEM) Toolbox to develop a suite of health management tools. The PEP includes functionality to perform the three types of prognostic models described previously in this guide. Each of the functions currently integrated in the PEP Toolbox are described below, categorized by function purpose. This list is continuously being updated as the PEP toolbox is developed and improved.

Prognostic Model Function Calls

initprog	Initialize prognostic model structure
runprog	Run prognostic model

Type I Models

initTypeI	Initialize a Type I (reliability-based) prognostic model
runTypeI	Run a Type I prognostic model

Type II Models

initMC	Initialize a Markov Chain prognostic model
MCprobs	Calculate the transition probability matrix and initial state probability vector
fitMC	Map operating conditions to system degradation
runMC	Run a Markov Chain prognostic model
initPHM	Initialize a Proportional Hazards prognostic model
runPHM	Run a Proportional Hazards prognostic model
initShock	Initialize a shock prognostic model
runShock	Run a shock prognostic model

Type III Models

initGPM	Initialize a General Path Model prognostic model
fitGPM	Determine GPM parametric model fit
initBayes	Calculate Bayesian prior distribution
threshGPM	Calculate GPM failure threshold
runGPM	Run a GPM prognostic model
initPF	Initialize Particle Filter model
fitPF	Determine PF parametric model fit
threshPF	Calculate PF failure threshold
initcoeff	Calculate PF coefficient prior distribution
runPF	Run PF prognostic model

Parameter Identification

optparam	Identify near-optimal prognostic parameter
paramfit	Prognostic parameter fitness
ppmetrics	Prognostic parameter suitability metrics
paramgen	Generate a population of prognostic parameters

Data Processing

MCdata	Discretize operating condition data
residgen	Generate monitoring system residuals using a PEM model
wtd_quantile	Calculate quantiles according to weighted inputs

PEP Functions Descriptions

fitGPM

Determine GPM parametric model fit

Syntax:

[Fit Ytransform] = FITGPM(prognosticparameters)

Description:

This function determines the best fit for a GPM prognostic model between linear, quadratic, and exponential fits.

[Function Ytransform] = FITGPM(prognosticparameters) determines the best fit for the historic paths contained in the cell array prognosticparameters. The function considers linear, quadratic, and exponential fits. Ytransform gives the transformation of y needed to make the fit linear in parameters. This has value @(y)log(y) for exponential functions and @(y)y (indicating no transformation is needed) for linear and quadratic functions.

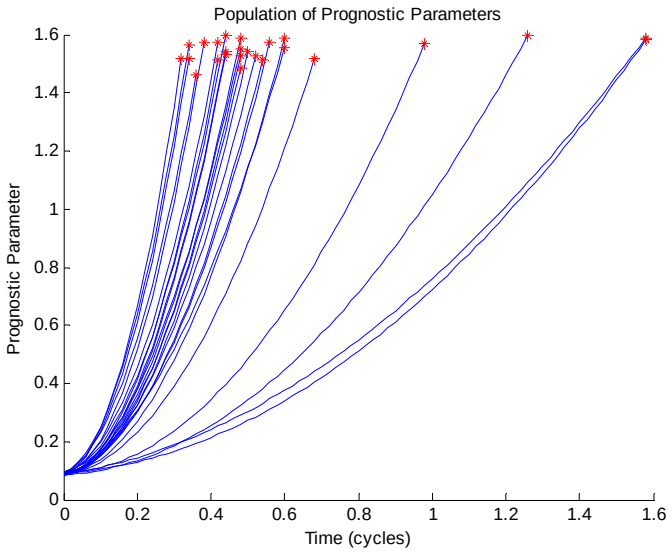
Example:

```
load degradation
whos
```

```
Name           Size            Bytes  Class     Attributes

  failed         1x25            15376  cell
  failtimes      1x5                40  double
```

```
  unfailed       1x5              1536  cell
```

```
failval = NaN(1,25);
failtime = NaN(1,25);
figure
hold on
for i = 1:25
    plot(failed{i}(:,1),failed{i}(:,2),'b')
    failval(i) = failed{i}(end,2);
   failtime(i) = failed{i}(end,1);
end
plot(failtime,failval,'r*')
hold off
xlabel('Time (cycles)')
ylabel('Prognostic Parameter')
title('Population of Prognostic Parameters')
```




```
[fit ytrans]=fitGPM(failed)
```

```
fit =

    @(x)x.^2    @(x)x    @(x)1

ytrans =

    @(y)y
```

fitMC

Map operating conditions to system degradation

Syntax:

b = FITMC(operatingconditions,model,threshold)

Description:

This function determines the coefficients to map time spent in each operating condition to system degradation.

b = MCFIT(operatingconditions,model,threshold) calculates the appropriate coefficients to fit the time spent in each operating condition, calculated from the operating conditions progressions to failure contained in the cell array operatingconditions, to the model (given by an anonymous function of the form @(b,t)f(b,t)) with ending value equal to threshold

Example:

```
load tire
whos
```

```
Name        Size              Bytes  Class    Attributes

  fail        1x100            297936  cell
  unfail      1x3                3436  cell
```

```
fit = @(b,t)t*b;
```

```
fitMC(fail,fit,100)
```

```
ans =

    0.1006

    0.2488

    0.5003
```

fitPF

Determine PF parametric model fit

Syntax:

[Function xtransform] = FITPF(prognosticparameters)

Description:

This function determines the best fit for a PF prognostic model between linear, quadratic, and exponential recursive fits.

[Function xtransform] = FITPF(prognosticparameters) determines the best fit for the historic paths contained in the cell array prognosticparameters. The function considers linear, quadratic, and exponential recursive fits. These "fits" are the additive portion of xt[x(n)] = xt[x(n-1)] + f(x(0:n-1),t(0:n)), where xt[] is the appropriate xtransform. This has value @(x)log(x) for exponential functions and @(x)x (indicating no transformation is needed) for linear and quadratic functions.

Example:

```
load PFdata
whos
```

```
  Name           Size            Bytes  Class     Attributes

  RUL_actual     1x50              400  double
  gtdata_trn     1x50            65600  cell
  gtdata_tst     1x50            65600  cell
  trn_data       1x50            38088  cell
  tst_data       1x50            30904  cell
```

```
[fit xtransform] = fitPF(trn_data)

fit =

    @(t,t1)t.^2-t1.^2    @(t,t1)t-t1


xtransform =

    @(x)x
```

initBayes

Calculate Bayesian prior distribution

Syntax:

[prior nvar] = initBayes(progparam,f,yt)

Description:

This function calculates the initial Bayesian prior distribution for a GPM model.

Prior = INITBAYES(prognosticparameters,fit,ytransform) calculates the Bayesian prior distribution for the coefficients, b, of the functional fit f(b,t) where fit is a cell array of the form f(x) = {@(x)f1(x) @(x)f2(x) ... @(x)fn(x)} and ytransform is a function handle P(y) = @(y)fy(y), where P(y) = f(x)*b for the historic prognostic parameter paths contained in the cell array prognosticparameters. It is assumed that the coefficients are normally distributed with mean and variance calculated from the population of fits. Prior is a matrix which contains the mean value for each parameter in the first row and the standard deviation in the second row:

$$prior = \begin{bmatrix} m_1 & \cdots & m_n \\ s_1 & \cdots & s_n \end{bmatrix}$$

Example:

```
load degradation
whos
```

```
  Name          Size            Bytes  Class     Attributes

  RULs          1x5                40  double
  failed        1x25            15376  cell
```

```
  unfailed      1x5             1536  cell
```

```
[fit, ytransform] = fitgpm(failed)
```

```
fit =

    @(x)x.^2    @(x)x    @(x)1


ytransform =

    @(y)y
```

```
 [prior noisevar] = initBayes(failed,fit,ytransform)
```

```
prior =

    6.1253    0.1396    0.0893
    3.4856    0.3286    0.0365


noisevar =

    0.0025
```

initCoeff

Calculate PF coefficient prior distribution

Syntax:

[Prior noisevar] = INITCOEFF(prognosticparameters,fit,xtransform)

Description:

This function calculates the initial Bayesian prior distribution for a PF model.

[Prior noisevar] = INITCOEFF(prognosticparameters,fit,xtransform) calculates the Bayesian prior distribution for the coefficients, b, of the functional fit f(b,t) where fit is a cell array of the form f(t,t1) = {@(t,t1)f1(t,t1) @(t,t1)f2(t,t1) ... @(t,t1)fn(t,t1)} and xtransform is a function handle P(x) = @(x)fx(x). x1 and t1 indicate the respective values at the previous observation, where P(x) = P(x1)+f(t,t1)*b for the historic prognostic parameter paths contained in the cell array prognosticparameters. It is assumed that the coefficients are normally distributed with mean and variance calculated from the population of fits. Prior is a matrix that contains the mean value for each parameter in the first row and the standard deviation in the second row:

$$prior - \begin{bmatrix} m_1 & \cdots & m_n \\ s_1 & \cdots & s_n \end{bmatrix}$$

noisevar is a scalar which gives an estimate of the variance of the measurement noise.

Example:

```
load PFdata
whos
```

```
  Name           Size            Bytes  Class     Attributes

  RUL_actual     1x50              400  double
  gtdata_trn     1x50            65600  cell
```

```
  gtdata_tst      1x50            65600  cell
  trn_data        1x50            38088  cell
  tst_data        1x50            30904  cell

[fit xtransform] = fitPF(trn_data);

[prior noisevar] = initCoeff(trn_data,fit,xtransform)

prior =

    0.0002    0.0002
    0.0000    0.0041


noisevar =

    0.0012
```

initGPM

Initialize a General Path Model prognostic model

Syntax:

model = initGPM(prognosticparameters)

model = initGPM(prognosticparameters,'flag',value,...)

Description:

This function initializes an GPM prognostic toolbox structure

Model = INITGPM(prognosticparameters) initializes a GPM model using the historic paths contained in the cell array prognosticparameters. Each cell should be an n x 1 matrix of prognostic parameter values, or an n x 2 matrix of time with parameters. The most appropriate fit is chosen from quadratic, linear, cubic, and exponential fits.

Model = INITGPM(prognosticparameters,'flag',value,...) initializes a GPM model using the historical prognostic parameters and the values indicated by the 'flag' - value pairs. 'flag' may be set to any of the following:

'bayesian' (false) Use Bayesian updating to include prior information in function fitting (true or false)

'noise' (calculated using enovar() in PEM) An estimate of the noise variance

'updateinterval' (1) Number of time steps between fitting updates (n, an integer) 'once' indicates the prior will be used once after all data is collected (in a non-dynamic way)

'fit' (optimization) Functional fit to use for GPM fitting (entered as a cell array of function handles, i.e. f = {@(x)f1(x) @x(f2(x) ... @(x)fn(x)} where P(x) = f(x)*B). Function must be linear in parameters for this type of entry. If a non-zero intercept is required, an "@(x)1" entry must be included. If no fit is given, an optimization is performed for the best fit among linear, quadratic, and exponential models.

'ytransform' (@(y)y) Functional transformation of y to make the prognostic parameter linear in parameters, i.e. if y = a*exp(bx) then log(y) = log(a)+b*x, entered as 'ytransform' = @(y)log(y) and 'fit' = {@(x)1 @(x)x}

'threshold' (calculated) Specify the critical threshold value (for hard thresholds) or the mean and standard deviation in the form [m s] (for threshold distributions)

'thresholdtype' ('pdf') Hard threshold or distribution ('hard'or 'pdf')

'threshcon' (0.95) The confidence level at which the threshold is calculated. Only applicable for "hard" type thresholds.

'npop' (1000) Number of Monte Carlo simulations used to make uncertainty estimates.

'allownegative' (false) Allow negative RUL estimates, false, true, or 'push'

Example:

```
load degradation
whos
```

```
Name          Size            Bytes  Class     Attributes

  RULs          1x5                40  double
  failed        1x25            15376  cell
  failtime      1x25              200  double
  failval       1x25              200  double
  i                 1x1                 8  double
  unfailed      1x5              1536  cell
```

```
model = initGPM(failed)


model =

             type: 'GPM'
         bayesian: 1
    updateinterval: 1
              fit: {[@(x)x.^2]  [@(x)x]  [@(x)1]}
       ytransform: @(y)y
        threshold: [1.5559 0.0514]
     thresholdtype: 'pdf'
         threshcon: 0.9500
              npop: 1000
         noisevar: 0.0025
     bayesianprior: [2x3 double]
     allownegative: 0
```

initMC

Initialize a Markov Chain prognostic model

Syntax:

Model = INITMC(operatingconditions)

Model = INITMC(operatingconditions,'flag',value,...)

Description:

This function initializes an MC model prognostics toolbox structure.

Model = INITMC(operatingconditions) initializes an MC model using the historic operating parameter progressions to failure contained in the cell array operatingconditions. These conditions should be discrete and numbered 1-n (no ordinal relation is implied by the numbering). The degradation parameter is assumed to be a weighted linear combination of the time spent in each operating condition. Initial degradation is assumed to be identically zero. The critical failure threshold for this parameter is assumed to be 100 (effectively measuring 100% degradation). If the data contained in operatingconditions is not numbered from 1 to n, the data may be reformatted in MCdata() prior to initializing the model.

Model = INITMC(operatingconditions,'flag',value,...) initializes an MC model using the historic operating condition progressions to failure and the user-supplied inputs indicated by the 'flag'-value pairs. 'flag' may be set to any of the following, with the default values given:

'Q' (calculated) The transition probability matrix for moving between operating conditions

'u' (calculated) The initial state probability vector

'f' (weighted sum of time spent in each condition) A function for mapping operating conditions to a degradation parameter. Should be input as an anonymous function @(b,t)f(b,t) where b is a column vector of coefficients and t is a row vector of time spent in each condition, 1-n

'b' (calculated from data) The vector of coefficients which is used with f to map the operating conditions to a degradation parameter

'threshold' (100)The critical failure threshold applied to the degradation parameter to indicate failure

'npop' (1000) The number of Monte Carlo simulations generated for each RUL estimation

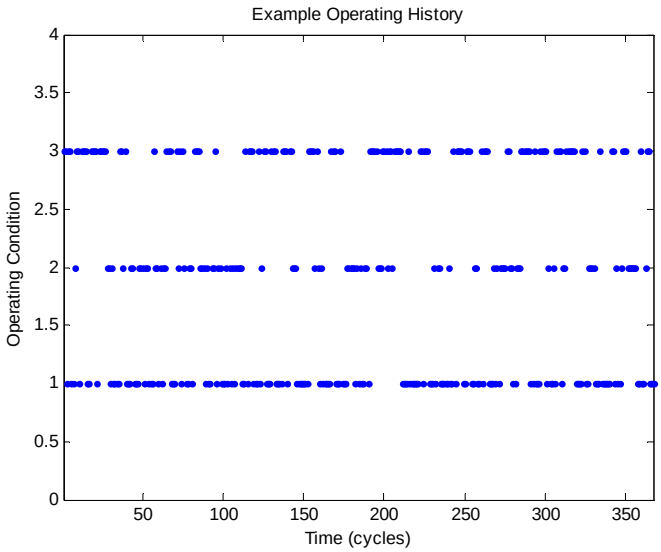
'RULcon' (0.95) The point in the resulting RUL distribution which defines the RUL. 0.95 indicates that RUL estimate is the point where the reliability of the system is 0.95 (Failure probability is 0.05)

Example:

```
load tire

whos
```

```
  Name        Size              Bytes  Class    Attributes


  fail        1x100            303136  cell
  unfail      1x3                3592  cell
```

```
figure

plot(fail{1},'.')

axis([-inf inf 0 4])

xlabel('Time (cycles)')

ylabel('Operating Condition')

title('Example Operating History')
```




```
model = initMC(fail)
```

```
model =

          type: 'MC'
             Q: [3x3 double]
             u: [0.2600 0.5500 0.1900]
             f: @(b,t)t*b
             b: [3x1 double]
     threshold: 100
          npop: 1000
        RULcon: 0.9500
```

```
model.Q

ans =

    0.4963    0.3092    0.1945
    0.4047    0.3935    0.2018
    0.2473    0.1540    0.5988

model.b

ans =

    0.1006
    0.2488
    0.5003
```

initPF

Initialize Particle Filter model

Syntax:

Model = INITPF(prognosticparameters)

Model = INITPF(prognosticparameters,'flag',value,...)

Description:

This funciton initializes a particle filter prognostic toolbox structure.

Model = INITPF(prognosticparameters) initializes a PF model using the historic paths contained in the cell array prognosticparameters. The most appropriate recursive fit is chosen from quadratic, linear, and exponential fits.

Model = INITPF(prognosticparameters,'flag',value,...) initializes a PF model using the historical prognostic parameters and the values indicated by the 'flag' - value pairs. 'flag' may be set to any of the following:

'bayesian' (true) Use Bayesian updating to adjust model parameter distributions when re-sampling

'noisevar' (calculated using enovar() in PEM) An estimate of the noise variance

'fit' (optimized) Functional fit to use for PF recursive fitting (entered as a cellarray of recursive function handles, i.e. f = {@(x,x1)f1(x,x1) @(x,x1)f2(x,x1) ... @(x,x1)fn(x,x1)} where x is the current observation and x1 is the previous observation and P(x) = f(x)*B) Function must be linear in parameters for this type of entry. If a non-zero intercept is required, an "@(x)1" entry must be included. If no fit is given, an optimization is performed for the best fit among recursive linear, quadratic, and exponential models

'transform' (@(y)y) Functional transformation of y to make the prognostic parameter linear in parameters, i.e. if the function is logarithmic, it can be considered as log(y) = log(y1)+b(x-x1), then ytransform = @(y)log(y) and fit = @(x,x1)x-x1

'modcoeff' (calculated) Distribution of model coefficients for sampling for particle forward prediction

'threshold' (calculated) Specify the critical threshold value (for hard thresholds) or the mean and standard deviation in the form [m s] (for threshold distributions)

'thresholdtype' ('pdf') Hard threshold or distribution ('hard' or 'pdf')

'threshcon' (0.95) The confidence level at which the threshold is calculated. Only applicable for "hard" type thresholds

'npart' (1000) Number of particles to simulate for each population

'particlethresh' (0.5) Fraction of effective particles which causes particle/coefficient resampling

'RULcon' (0.95) The point in the resulting RULdistribution which defines the RUL. 0.95 indicates that RUL estimate is the point where the reliability of the system is 0.95 (Failure probability is 0.05)

Example:

```
load PFdata
whos
```

```
  Name            Size              Bytes  Class     Attributes

  RUL_actual      1x50                400  double
  gtdata_trn      1x50              65600  cell
  gtdata_tst      1x50              65600  cell
  trn_data        1x50              38088  cell
  tst_data        1x50              30904  cell
```

```
model = initPF(trn_data)
```

```
model =

              type: 'PF'
               fit: {[@(t,t1)t.^2-t1.^2]  [@(t,t1)t-t1]}
         transform: @(x)x
          bayesian: 1
          modcoeff: [2x2 double]
    particlethresh: 0.5000
     thresholdtype: 'pdf'
         threshold: [1.5492 0.1044]
          noisevar: 0.0012
             npart: 1000
            RULcon: 0.5000
```

initPHM

Initialize a Proportional Hazards prognostic model

Syntax:

Model = INITPHM(cov,times)

Model = INITPHM(cov,times,'flag',value...)

Description:

This function initializes a PHM prognostics toolbox structure.

Model = INITPHM(cov,times) initializes a proportional hazards model with the covariates defined in the nxp matrix cov and the failure times in the nx1 column matrix times. The baseline value is taken to be zero.

Model = INITPHM(cov,times,'flag',value...) initializes a proportional hazards model using the covariates and failures times in the matrics cov and times, and the 'flag'-value pairs given. 'flag' may be set to any of the following, with the default values given:

'frequency' (one for each observation) The jth element of this nx1 column vector indicates the number of times that the combination of the jth set of covariates and the jth failure time are observed together.

'censoring' (all zeros) The jth element of this nx1 column vector indicates whether that observation is a failure time (0) or a censored time (1).

'baseline' (zero) This 1xp row vector indicates the baseline values for each of the covariates.

'beta' (calculated) The coefficients of the Cox proportional hazards regression.

'hazard' (calculated) The baseline cumulative hazard function given as a 2-column matrix with failure times in the first column and the corresponding estimated cumulative hazard rate in the second column.

'RULcon' (0.95)The point in the resulting RUL distribution which defines the RUL. 0.95 indicates that RUL estimate is the point where the reliability of the system is 0.95 (Failure probability is 0.05).

Example:

```
load PHMchalldata

whos
```

```
  Name        Size                Bytes  Class     Attributes

  RUL       259x1                  2072  double
  trn         1x260             10350848  cell
  tst         1x259              6555280  cell
```

```
trn_cov = cell(size(trn));

failtimes = NaN(numel(trn),1);

for i = 1:length(trn)

   trn_cov{i} = trn{i}(:,1:3);

   failtimes(i) = length(trn{i});

end
```

```
model = initPHM(trn_cov,failtimes)
```

```
model =

        type: 'PHM'
        beta: [3x1 double]
      hazard: [261x2 double]
```

```
baseline: [0 0 0]

  RULcon: 0.9500
```

initprog

Initialize prognostic model structure

Syntax:

Model = INITPROG('type',...)

Model = INITPROG('typeI',TTF)

Model = INITPROG('typeI',TTF,censored)

Model = INITPROG('typeI',TTF,censored,'distribution',distributiontype)

Model = INITPROG('mc',operatingconditions)

Model = INITPROG('mc',operatingconditions,'flag',value,...)

Model = INITPROG('shock',...)

Model = INITPROG('phm',...)

Model = INITPROG('gpm',prognosticparameters)

Model = INITPROG('gpm',prognosticparameters,'flag',value...)

Model = INITPROG(‘pf’,prognosticparameters)

Model = INITPROG(‘pf’,prognosticparameters,’flag’,value…)

Description:

This function initializes a Prognostics Toolbox model structure.

Model = INITPROG('type',...) initializes a prognostic model of the kind specified by 'type'. 'type' may be set to any of the following strings:

'typeI'

'markovchain' (or 'mc')

'shock'

'proportionalhazards' (or 'phm')

'generalpath' (or 'gpm')

‘particlefilter’ (or ‘pf’)

For necessary inputs and available flag/value options, please see the appropriate individual “init” functions: initTypeI, initMC, initShock, initPHM, initGPM, initPF.

Example:

For examples of each model type, please see the appropriate init functions, as listed above. To utilize the initprog wrapper function, simply indicate the appropriate type as the first input.

initShock

Initialize a shock prognostic model

Syntax:

Model = INITSHOCK(shocks)

Model = INITSHOCK(shocks,'flag',value,...)

Description:

This function initializes a shock model prognostics toolbox structure

Model = INITSHOCK(shocks) initializes a shock model using the readings (such as vibration or current) which contain a progression of random shocks contained in the cell array shocks. Each cell may contain a vector of the measurement of interest or an nX2 matrix where the first column is time and the second column is the sensed value. Here, the indication of failure is the sum of the shock sizes.

Model = INITSHOCK(shocks,'flag',value,...) initializes a shock model using the progression of random shocks contained in the cell array shocks and the values indicated by the 'flag' - value pairs. 'flag' may be set to any of the following:

'operatingcond' (none) A cell array of column vectors that indicate the operating conditions for each shock progression. The operating condition at each time step should be an integer from 1 to n, which indicates which of n discrete operating conditions the system is in. This can account for the difference in probability of a shock occurring for each operating condition. This results in a Type II model. Note that this option CANNOT be used with a degradation dependant distribution. The function MCdata() can be used to put operating conditions into discrete groups numbered 1 to n. Indicated in model structure as model.TypeII = true.

'degdepend' (false) Parameter of the time distribution can be dependent on the degradation level, as might happen in a vibrating bearing or failing power supply. This effectively makes the shock model a Type III model. Lambda is assumed to depend on the degradation according to a polynomial of degree 2 unless otherwise specified by 'timeorder'. A non-polynomial function can be specified with 'timefcn'. Note that this option CANNOT be used with operating condition information. Indicated in model structure as model.TypeIII = true.

'degcondition' Measurements of equipment degradation can be used to develop a degradation dependent shock model. Input should be a cell array of degradation measures of the same size as shocks.

'timeorder' (2) The order of polynomial used to describe the dependence on degradation.

'timefcn' (polynomial) An anonymous function of the form @(d)f(d) for degradation-dependent distributions. For instance, if the distribution is dependent on the exponential function, then the timefcn would be input as: @(b,d)exp(b(1)*d+b(2)) The parameters in b will be fit to the available data. Notice that your function parameters must be a vector called 'b' to be fit. If they are known a priori, they can be specified with 'timecoef'.

'timecoef' (fit) A vector which contains the coefficients of the degradation dependent function used to define the time distribution.

'timepar' (distribution fit) Parameter of a static (not degradation dependent) time distribution, which is assumed to be exponential. This is the lambda parameter of the exponential distribution.

'magdist' (normal) Distribution of the magnitude of shocks ('constant', 'normal', 'nonparametric' or 'np')

'magpar' (distribution fit) Parameters of magnitude distribution

'minshock' (calculated) The minimum size of the input that is considered a shock. It is calculated as the mean non-shock input value plus three standard deviations of the noise.

'npop' (1000) The number of Monte Carlo simulations generated for each RUL estimation

'RULcon' (0.50) The point in the resulting RULdistribution which defines the RUL. 0.5 indicates that RUL estimate is the point where the reliability of the system is 0.5 (Failure probability is 0.5)

'thresholdtype' ('pdf') Hard threshold or distribution ('hard' or 'pdf')

'threshold' (calculation) Specify the critical threshold value (for hard thresholds) or the mean and standard deviation in the form [m s] (for threshold distributions)

'threshcon' (0.50) Confidence level for determining threshold.

Example:

```
load shockdata
whos
  Name          Size              Bytes  Class     Attributes
  shocks        1x100            173376  cell
  test        193x1                1544  double

model = initshock(shocks,'degdepend',true,'timeorder',1)
model =

            type: 'Shock'
         minshock: 3.0263
           TypeII: 0
          TypeIII: 1
          timepar: [1x1 struct]
          magdist: 'normal'
           magpar: [1x1 struct]
             npop: 1000
    thresholdtype: 'pdf'
        threshold: [1x1 struct]
        threshcon: 0.5000
           RULcon: 0.5000
```

initTypeI

Initialize a Type I (reliability-based) prognostic model

Syntax:

Model = INITTYPEI(TTF)

Model = INITTYPEI(TTF,censored)

Model = INITTYPEI(TTF,censored,'distribution',distributiontype)

Description:

This function initializes a reliability-based prognostic model structure.

Model = INITTYPEI(TTF) initializes a reliability-based prognostic model using the failure times in the vector TTF. This model uses a Weibull distribution to model failure times.

Model = INITTYPEI(TTF,censored) initializes a model using the failure and censoring times in the vector TTF. The variable censored indicates whether each observed time is a failure (0) or censored (1) time.

Model = INITTYPEI(TTF,censored,'distribution',distributiontype) initializes a Type I model using the failure time distribution indicated by distributiontype. distributiontype can be set to any of the following strings: 'weibull', 'exponential' (or 'exp'), 'normal'. The default distribution type is Weibull. Note that the variable censored may be omitted if all data points are actual failure times.

Example:

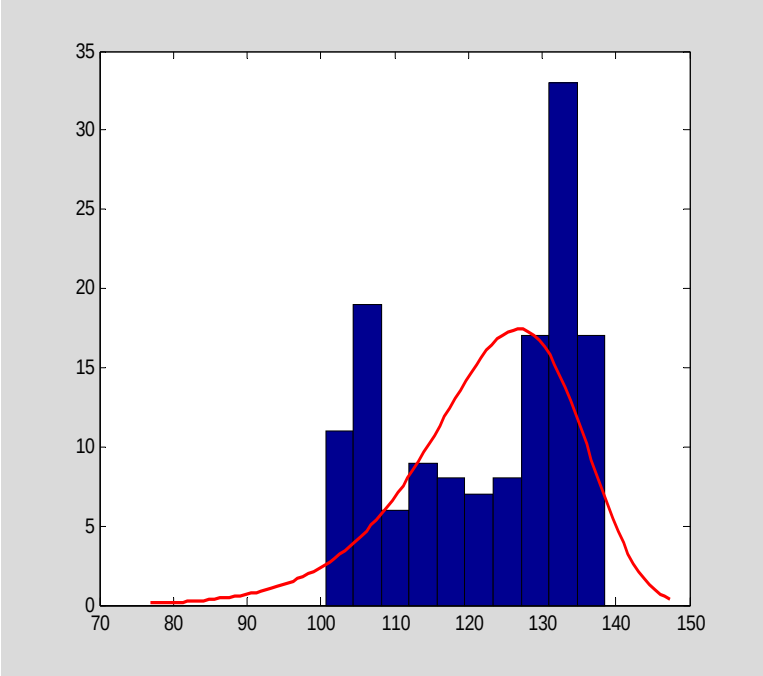
```
load failuretimes
whos
```

```
  Name                Size              Bytes  Class     Attributes
```

```
  failuretimes      135x1                1080  double
```

```
typeI = inittypeI(failuretimes)

typeI.parameters

histfit(failuretimes,10,'wbl')
```

```
typeI =

            type: [1x5 char]
    distribution: [1x7 char]
      parameters: [1x1 struct]
            data: [1x1 struct]

ans =

     beta: 13.0851
    theta: 127.5712
```



MCData

Discretize operating condition data

Syntax:

[newoperatingconditions map] = MCDATA(operatingconditions)

[newoperatingconditions map] = MCDATA(operatingconditions, 'flag', value)

Description:

This function converts operating condition data into data needed for Markov Chain Models

[newoperatingconditions map] = MCDATA(operatingconditions) converts the operating condition progressions to failure contained in the cell array operatingconditions into conditions numbered 1-n (no ordinal relation is implied by the new numbers). The progressions in the cell array operatingconditions may be of size nxm where n is the number of observations in one history and m is the number of variables which fully define the operating condition. These values need to be discrete (or discretized by some outside method) to apply this function correctly. The output map is a matrix which indicates the relationship between the original operating conditions (of size 1xn) to an operating condition class. The first row in map defines the first operating condition, the second row, the second operating condition, and so on.

[newoperatingconditions map] = MCDATA(operatingconditions, 'flag', value) separates the operating conditions progressions contained in the cell array operatingconditions into MC appropriate conditions. The 'flag'/value pairs may be set to any of the following (defaults):

'tol' (0.10) The noise tolerance for separating operating conditions.

'map' (determined) The map for moving from measured operating conditions to the MC numbered conditions where the first row in map defines the first operating condition, the second row in map defines the second operating condition, and so on.

Example:

```
load PHMchalldata

whos
```

```
  Name        Size                Bytes  Class     Attributes

  RUL       259x1                  2072  double
  trn         1x260             10337328  cell
  tst         1x259              6541812  cell
```

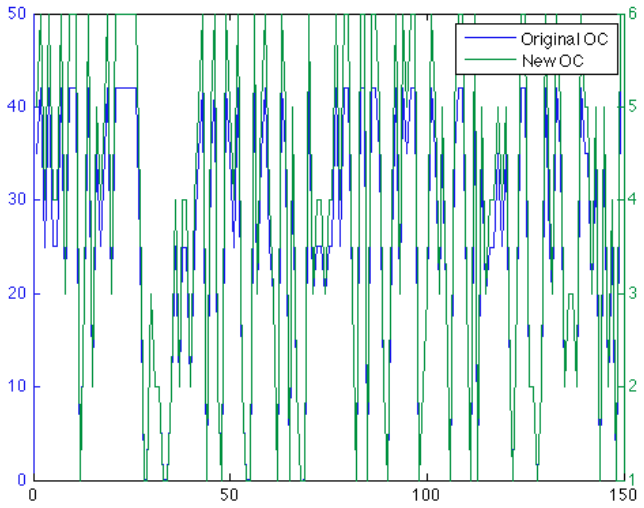
```
old_oc = cell(size(trn));
for i = 1:length(trn)
    old_oc{i} = trn{i}(:,1);
end
```

```
[new_oc map] = MCdata(old_oc);
```

```
map
```

```
map =

    0.0030
   10.0080
   20.0080
   25.0080
   35.0080
   42.0080
```

```
t = 1:length(old_oc{1});

figure

plotyy(t,old_oc{1},t,new_oc{1})

legend('Original OC','New OC')
```



MCprobs

Calculate the transition probability matrix and initial state probability vector

Syntax:

[Q u] = MCPROBS(operatingconditions)

Description:

MCPROBS calculates the transition and initial condition probabilities. This function calculates the transition probability matrix and the initial condition probability vector for a Type II Markov Chain model

[Q u] = MCPROBS(operatingconditions) calculates the transition probability matrix (Q) and the initial condition vector (u) for a Type II Markov Chain model using the operating condition progressions to failure contained in the cell array operatingconditions.

Example:

```
load PHMchalldata
whos
```

```
  Name        Size                Bytes  Class     Attributes

  RUL       259x1                  2072  double
  trn         1x260             10337328  cell
  tst         1x259              6541812  cell
```

```
old_oc = cell(size(trn));
for i = 1:length(trn)
```

```
    old_oc{i} = trn{i}(:,1);
end

[new_oc map] = MCdata(old_oc);

[Q u] = MCprobs(new_oc)

Q =

    0.1474    0.1520    0.1512    0.1521    0.1430    0.2544
    0.1440    0.1508    0.1502    0.1472    0.1524    0.2554
    0.1455    0.1466    0.1517    0.1468    0.1588    0.2506
    0.1543    0.1568    0.1479    0.1507    0.1415    0.2488
    0.1502    0.1551    0.1542    0.1465    0.1511    0.2431
    0.1536    0.1460    0.1518    0.1488    0.1500    0.2498


u =

    0.1577      0.1346      0.1231      0.1885      0.1385      0.2577
```

optparam

Identify near-optimal prognostic parameter

Syntax:

param = OPTPARAM(inputs, 'flag', value, ...)

Description:

This function uses Genetic Algorithms and parameter suitability metrics to identify a near-optimal prognostic parameter from several data sources.

param = OPTPARAM(inputs, 'flag', value, ...) identifies a near-optimal prognostic parameter from the available data in the cell array inputs. The function uses genetic algorithms to optimize the weights in a linear combination of the available signal inputs. Function returns a structure, param, which includes the resulting weights and all information needed to obtain the parameter from a new data run using function PARAMGEN. Flag/value pairs may be set to any of the following:

'inputs' ('all'): Determines how many of the inputs should be considered by the GA. Can be set to 'all', indicating that all should be used, or 'subset', indicating that a subset of useful parameters should be used.

'cutoff' (1.5): Cutoff value for determining which input parameters are useful when choosing a subset of inputs for optimization.

'fitness' (sum of M,P,T): Identifies the fitness function to be used, input as @fitness. fitness(w,inputs) must be a matlab m-file which takes only the candidate solution from the GA and the cell array of possible inputs to determine the fitness of a candidate solution.

'fitweights' ([1 1 1]): A 1x3 row vector of weights which gives the weight of each of the three suitability metrics in determining the fitness. The first entry corresponds to monotonicity, the second to prognosability, and the third to trendability.

'initpop' ([]): Any candidate weightings that the GA should consider. If visual inspection or expert analysis has lead to any suitable parameters, these can be included in the GA to allow it to explicitly consider them in the optimization. They should be included as an nXm matrix where n is the number of possible prognostic parameters to be included and m is the number of candidate

inputs to the parameter. Each row should contain the weights associated with one possible parameter.

'group' (one group): Multiple prognostic parameters may be optimized to compare groups. Associated value must be a cell array of matrices where each matrix indicates the runs included in a specific group. Groups may be overlapping. Output in this case will be a structure array where the ith entry corresponds to the optimum parameter for the ith group.

'smoothing' (false): May be set to 'true' or 'false', indicates whether the resulting parameter should be smoothed prior to final model development.

Example:

```
load PHMchalldata_monitoringresults

whos

  Name      Size                Bytes  Class    Attributes

  Fhyp      1x260             9060632  cell
  pred      1x260             9060632  cell
  res       1x260             9060632  cell

param = optparam(res,'inputs','subset','cutoff',1.5)

param =

      weights: [0.2497 0.1287 0.2514 5.0065 3.1495 0.4695]
       inputs: [4 5 6 13 16 18]
```

```
   cutoff: 2
  fitness: @(x)param.fitness(x,inputs)
smoothing: 0
```

paramfit

Prognostic parameter fitness

Syntax:

fitness = PARAMFIT(weights,inputs)

fitness = PARAMFIT(weights,inputs,suitabilityweights)

Description:

PARAMFIT determines the fitness of a candidate prognostic parameter. This function determines the fitness of a candidate prognostic parameter by calculating the sum of monotonicity, prognosability, and trendability. These three prognostics metrics are computed using the function PPMETRICS.

fitness = PARAMFIT(weights,inputs) calculates the fitness of a linear combination of inputs, weighted according to the row vector weight, as the sum of the parameter suitability metrics monotonicity, prognosability, and trendability.

fitness = PARAMFIT(weights,inputs,suitabilityweights) calculates the fitness of a linear combination of inputs, weighted according to the row vector weight, as the weighted sum of the parameter suitability metrics monotonicity, prognosability, and trendability, where suitabilityweights is a 1x3 row vector with the weight of monotonicity in the first entry, prognosability in the second, and trendability in the third.

fitness = PARAMFIT(weights,inputs,suitabilityweights,filter) filters the resulting prognostic parameter before calculating the suitability. An exponential filter is used via expfilt().

Example:

```
load PHMchalldata_monitoringresults

whos

  Name      Size                Bytes  Class    Attributes
```

```
  Fhyp      1x260            9060632  cell
  pred      1x260            9060632  cell
  res       1x260            9060632  cell

fitness = paramfit(ones(1,21),res)

fitness =

   -1.0485
```

paramgen

Generate a population of prognostic parameters

Syntax:

progparam = PARAMGEN(param_struct,inputs)

progparam = PARAMGEN(param_struct,inputs,time)

Description:

This function generates a population of prognostic parameters according to the options saved in the structure param_struct.

progparam = PARAMGEN(param_struct,inputs) creates a population of prognostic parameters using the options saved in param_struct and the candidate inputs in the cell array inputs. OPTPARAM() can be used to generate the parameter structure.

progparam = PARAMGEN(param_struct,inputs,time) creates a population of prognostic parameters which includes the time variable for each case in the first column and the prognostic parameter in the second. Both inputs and time should be cell arrays where the number of observations (rows) in each cell of inputs should be equal to the corresponding cell of time. PARAMGEN calls also produce a plot of the population of prognostic parameters.

progparam = PARAMGEN(...,plot) plots resulting prognostic parameters if set to true. Default is false.

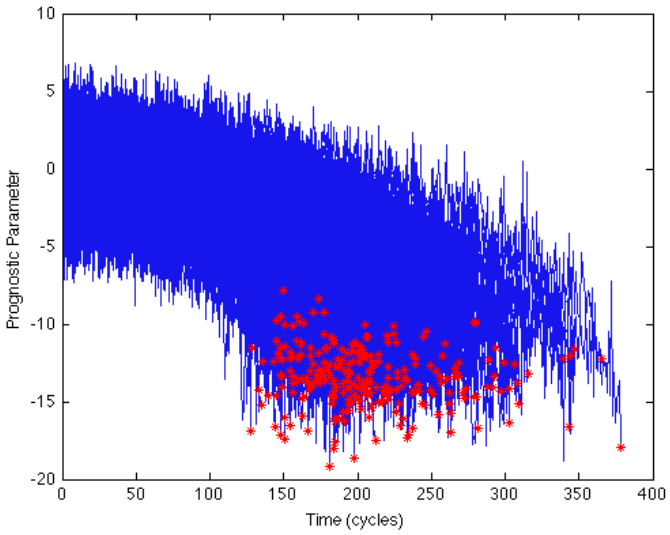
Example:

```
load PHMchalldata_ga

load PHMchalldata_monitoringresults

whos
```

```
  Name       Size                Bytes  Class     Attributes
```

```
  Fhyp        1x260            9060632  cell
  m           1x1                    8  double
  p           1x1                    8  double
  par         1x260             459192  cell
  param       1x1                 1017  struct
  pred        1x260            9060632  cell
  res         1x260            9060632  cell
  t           1x1                    8  double
```

```
prognosticparameter = paramgen(param,res,true);
```



ppmetrics

Prognostic parameter suitability metrics

Syntax:

[monotonicity prognosability trendability] = PPMETRICS(params)

Description:

This function characterizes the appropriateness of a prognostic parameter based on three metrics.

[monotonicity prognosability trendability] = PPMETRICS(params) evaluates the population of prognostic parameters contained in the cell array params for three metrics of adequacy.

Montonicity measures the general increasing or decreasing trend of the parameter. Because the assumption is made that systems do not self heal and no corrective action is taken, prognostic parameters are assumed to be monotonic. This assumption may not be valid for some systems such as batteries that do exhibit some self healing during periods of rest, or systems which experience some outside intervention to improve the condition.

Prognosability is a measure of the variance of the failure values for a population of parameters.

Trendability characterizes how well a population of parameters can be fit by the same functional form. It measures the similarity of the general trend of the parameter for a population of systems.

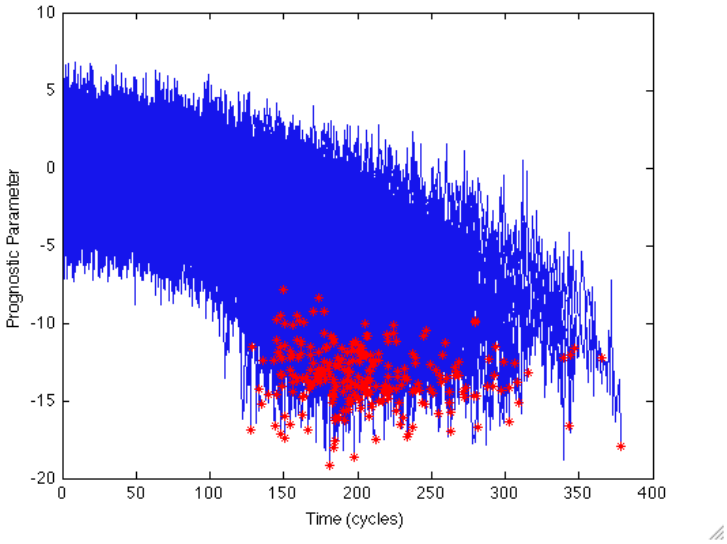
Example:

```
load PHMchalldata_ga

load PHMchalldata_monitoringresults

whos
```

```
  Name       Size                Bytes  Class     Attributes

  Fhyp       1x260             9060632  cell
```

```
  m          1x1                     8  double
  p          1x1                     8  double
  par        1x260              459192  cell
  param      1x1                  1017  struct
  pred       1x260             9060632  cell
  res        1x260             9060632  cell
  t          1x1                     8  double
```

```
prognosticparameter = paramgen(param,res,true);
```




```
[m p t] = ppmetrics(prognosticparameter)
```

```
m =

    0.6916

p =

    0.8674

t =

    0.8684
```

residgen

Generate monitoring system residuals using a PEM model

Syntax:

RESID = RESIDGEN(model,data)

Description:

This function uses a PEM model to calculate the residuals of a population of systems.

RESID = RESIDGEN(model,data) generates residuals using the previously developed PEM toolbox model and a cell array of data for a population of systems. Note that the generated residuals are predictions - actual.

Example:

```
load PHMchalldata
load PHMchalldata_models
whos
```

```
  Name           Size                Bytes  Class     Attributes

  Groups         1x2                   456  cell
  RUL          259x1                  2072  double
  model          1x1                1055382  struct
  test        1986x21                333648  double
  train       3972x21                667296  double
  trn            1x260             10350848  cell
  tst            1x259              6555280  cell
```

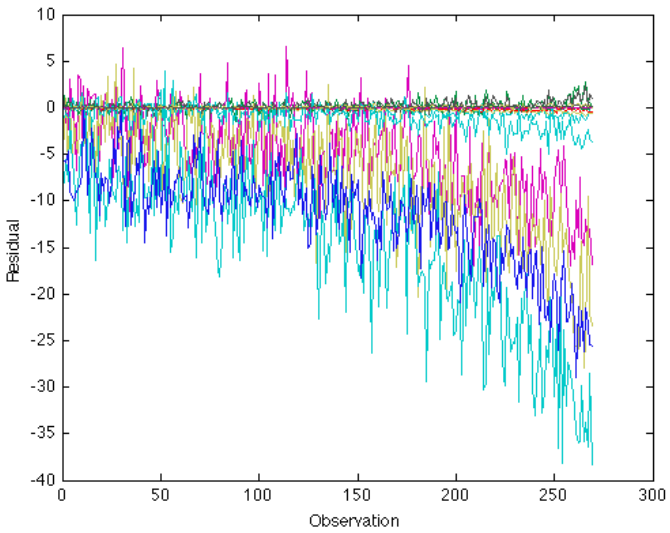
```
  val         1987x21              333816  double
```

```
trn_g1 = cell(size(trn));
for i = 1:length(trn)
   trn_g1{i} = trn{i}(:,Groups{1});
end
```

```
resid = residgen(model,trn_g1);
```

```
whos
  Name           Size                 Bytes  Class     Attributes

  Groups         1x2                    456  cell
  RUL          259x1                   2072  double
  i              1x1                      8  double
  model          1x1                1055382  struct
  resid          1x260              9060632  cell
  test        1986x21                333648  double
  train       3972x21                667296  double
  trn            1x260             10350848  cell
  trn_g1         1x260              9060632  cell
  tst            1x259              6555280  cell
  val         1987x21                333816  double
```

```
figure

plot(resid{1})

xlabel('Observation')

ylabel('Residual')
```



runGPM

Run a GPM prognostic model

Syntax:

RUL = RUNGPM(model,currentparam)

RUL = RUNGPM(model,timestamp,currentparam)

Description:

This function makes RUL estimates using a general path model.

RUL = RUNGPM(model,currentparam) makes prognostic estimates using a Type III model. 'model' should be of type 'gpm'. 'currentparam' may be a column vector, matrix, or cell array. If only one system is under surveillance, currentparam should be a column vector containing observations of that system up to the current time. If multiple systems are under surveillance, and parameter observations are available for the same time steps for each system, currentparam may be a matrix whose columns contain the parameter observations for a single system. If multiple systems are under surveillance which have been running for different amounts of time, currentparam may be a cell array containing the parameter observations for each system in a column vector contained in separate cells. Here, it is assumed that observations are made every time unit, with an equal sampling interval. Each cell is an n x 1 matrix of parameters at equal time intervals or an n x 2 matrix of time and parameters.

RUL = RUNGPM(model,timestamp,currentparam) can be used when the sampling interval is not equal across observations. If currentparam is a column vector, then timestamp should also be a column vector. If currentparam is a matrix, then timestamp may be a column vector of times (if each unit is surveyed at the same time) or it may be a matrix of times (if surveillance times for each unit are different).

RUL = RUNGPM(model,currentparam,typeImodel,typeIImodel) takes in a Type I and/or Type II MC model to be used as priors for the GPM. The prior models are standard PEP generated prognostic structures. RUNGPM(model, currentparam, phmmodel, covariates) if a PHM model is used, immediately after input a cases x 1 array of covariates

RUL = RUNGPM(...) returns a structure of the estimated RUL, the Monte Carlo standard deviation, the associated 95% confidence intervals, and the model structure with an updated Bayesian prior for future estimates. No other fields are changed.

Example:

```
load degradation

whos
  Name          Size            Bytes  Class     Attributes

  RULs          1x5                40  double
  failed        1x25            15376  cell
  unfailed      1x5              1536  cell

model = initprog('gpm',failed)

model =

            type: 'GPM'
        bayesian: 0
             fit: {[@(x)x.^3]  [@(x)x.^2]  [@(x)x]  [@(x)1]}
      ytransform: @(y)y
        fiterror: 1.0582
       threshold: [10.0134 0.5691]
```

```
    thresholdtype: 'pdf'
        threshcon: 0.9500
             npop: 1000
    allownegative: 0
```

```
RUL = runGPM(model,unfailed)
```

```
           RUL: 51.3636
        StdDev: 113.4787
          CI95: [0 352.7522]
    Parameters: [6.9446e-007 -1.5391e-004 0.0205 1.0690]
    ParmStdDev: [4.0355e-014 6.2635e-009 7.7975e-005 0.0700]
        Priors: {[]}
```

runMC

Run a Markov Chain prognostic model

Syntax:

RUL = RUNMC(model,operatingconditions)

RUL = RUNMC(model,'new')

[RUL uncert] = RUNMC(…)

Description:

This function makes RUL estimates using a Type II MC model

RUL = RUNMC(model,operatingconditions) makes prognostic estimates using a Type II Markov Chain model. model should be of type 'MC'. operating conditions is a column vector of the operating conditions seen by the unit under test up to the current time. If operatingconditions is a cell array of column vectors, one vector for the operation of a single unit to the current time, then RUL is a row vector of RUL times.

RUL = RUNMC(model,'new') calculates the expected RUL of a system starting from new, with no information about its operating states.

[RUL uncert] = RUNMC(...) returns the RUL prediction and the associated 95% uncertainty interval.

Example:

```
load tire
whos
```

```
  Name        Size              Bytes  Class    Attributes

  fail        1x100            303136  cell
```

```
  unfail      1x3                3592  cell
```

```
model = initMC(fail);
```

```
RUL = runMC(model,unfail)
```

```
              RUL: [3x1 double]
           StdDev: [3x1 double]
             CI95: [3x2 double]
    MonteCarloRUL:                    [3x1000                    double]
```

runPF

Run PF prognostic model

Syntax:

RUL = RUNPF(model ,currentparam)

RUL = RUNPF(model,timestamp,currentparam)

[RUL uncert] = RUNPF(…)

Description

This function makes RUL estimates using a particle filtering model.

RUL = RUNPF(model,currentparam) makes prognostic estimates using a particle filtering model. The model should be of type 'PF'. currentparam may be a column vector, matrix, or cell array. If only one system is under surveillance, currentparam should be a column vector containing observations of that system up to the current time. If multiple systems are under surveillance, and parameter observations are available for the same time steps for each system, currentparam may be a matrix whose columns contain the parameter observations for a single system. If multiple systems are under surveillance which have been running for different amounts of time, may be a cell array containing the parameter observations for each system in a column vector contained in separate cells. Here, it is assumed that observations are made every time unit, with an equal sampling interval.

RUL = RUNPF(model,currentparam) may also be used for multiple systems under surveillance where the time stamp for each system is not the same. In this case, currentparam should be a cell array whose cells contain an nx2 matrix where the first column is the time stamp for that particular unit and the second column is the parameter values at each time.

RUL = RUNPF(model,timestamp,currentparam) can be used when the sampling interval is not equal across observations. If currentparam is a column vector, then timestamp should also be a column vector. If currentparam is a matrix, then timestamp may be a column vector of times (if each unit is surveyed at the same time) or it may be a matrix of times (if surveillance times for each unit are different).

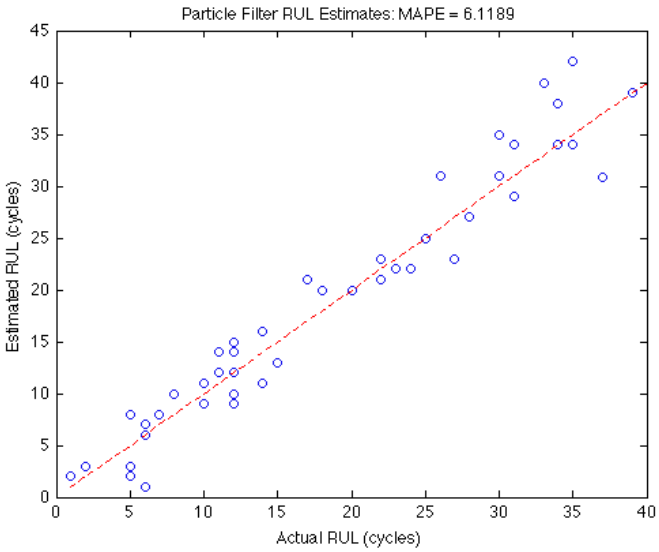
[RUL uncert] = RUNPF(...) returns the estimated RUL(s) and the associated 95% uncertainty interval(s).

Example:

```
load PFdata

whos
```

```
  Name            Size            Bytes  Class     Attributes


  RUL_actual      1x50              400  double
  gtdata_trn      1x50            65600  cell
  gtdata_tst      1x50            65600  cell
  trn_data        1x50            38088  cell
  tst_data        1x50            30904  cell
```

```
model = initPF(trn_data);

[RUL uncert] = runPF(model,tst_data);


figure; hold on

plot(RUL_actual,RUL,'o')

plot(1:40,1:40,'r--')

xlabel('Actual RUL (cycles)')

ylabel('Estimated RUL (cycles)')

title(['Particle Filter RUL Estimates: MAPE = ' num2str(mean((RUL-
RUL_actual')./RUL_actual'*100))])
```


Particle Filter RUL Estimates: MAPE = 6.1189
Estimated RUL (cycles)
Actual RUL (cycles)
0
5
10
15
20
25
30
35
40
45

runPHM

Run a Proportional Hazards prognostic model

Syntax:

[RUL uncert] = RUNPHM(model,covariates)

Description:

This function makes RUL estimates using a Proportional Hazards Model.

[RUL uncert] = RUNPHM(model,currentsystem) makes prognostic estimates using a Proportional Hazards model. currentsystem is an nx2 matrix where the first column is the current time for a particular system and the second column is the covariate value for that system. The output RUL is an nx1 matrix of RUL estimates based on the reliability confidence level and uncert is an nx2 matrix of 95% uncertainty intervals.

Example:

```
load PHdata
whos
```

```
  Name              Size            Bytes  Class     Attributes

  baseline          1x1                 8  double
  censoring        16x1               128  double
  covariates       16x1               128  double
  failuretimes     16x1               128  double
  frequency        16x1               128  double
```

```
model = initPHM(covariates,failuretimes,'baseline',baseline,...
```

```
    'frequency',frequency,'censoring',censoring);


covariate = 1/(170+273);

times = 0:500:2000;

currentsystem = [times' covariate*ones(5,1)];

[RUL uncert] = runPHM(model,currentsystem)


RUL =


     RUL: [5x1 double]

    CI95: [5x2 double]
```

runprog

Run prognostic model

Syntax:

RUL = RUNPROG(model,...)

RUL = RUNPROG(model,currenttime)

RUL = RUNPROG(model)

RUL = RUNPROG(model,currentenvir)

RUL = RUNPHM(model,covariates)

RUL = RUNPROG(model,currentparam)

Description:

This function estimates the Remaining Useful Life (RUL) using the model previously developed.

For necessary inputs, please see the appropriate individual "run" functions: runTypeI, runMC, runShock, runPHM, runGPM, runPF.

Example:

For examples of each model type, please see the appropriate *run* functions, as listed above. To utilize the runprog wrapper function, provide an initialized model of the appropriate model type.

runShock

Run a shock prognostic model

Syntax:

RUL = RUNSHOCK(model,shocks)

RUL = RUNSHOCK(model,shocks,operatingconditions)

RUL = RUNSHOCK(model,shocks,condition)

[RUL uncert] = RUNSHOCK(…)

Description:

This function makes RUL estimates using a Shock Model.

RUL = RUNSHOCK(model,shocks) makes prognostic estimates using a Type II shock model. shocks is a vector of measurements, such as vibration or voltage, which indicate the measure of interest. shocks may be an nx2 matrix where the first column is the time stamp and the second column is the sensor measurement.

RUL = RUNSHOCK(model,shocks,operatingconditions) considers the planned future operating condition in generating random shocks. operatingconditions may be a column vector of discrete condition indicators for possible future time, an nx2 matrix where the first column is the planned time steps and the second is the operating condition, or a single scalar indicating the operating condition for all future use.

RUL = RUNSHOCK(model,shocks) makes prognostic estimates using a Type III shock model. shocks is a vector of measurements, such as vibration or voltage, which indicate the measure of interest. In this case, model includes shock probabilities dependent on the current estimated equipment condition.

RUL = RUNSHOCK(model,shocks,condition) makes prognostic estimates using a Type III shock model and the current measured equipment condition.

[RUL uncert] = RUNSHOCK(...) returns RUL predictions and the associated 95% uncertainty interval.

Example:

```
load shockdata
whos
  Name          Size              Bytes  Class     Attributes
  shocks        1x100            173376  cell
  test        193x1                1544  double

model = initshock(shocks,'degdepend',true,'timeorder',1);
t = [39 77 116 154];
RUL = NaN(4,1);
uncert = NaN(4,2);
for i = 1:4
   [RUL(i) uncert(i,:)] = runShock(model,test(1:t(i)));
end

RUL =

   78.0986
   61.5199
   38.5305
   36.3906
```

```
uncert =

   16.4980  210.4872
    4.6990  184.0803
         0  140.0638
         0  148.2056
```

runTypeI

Run a Type I prognostic model

Syntax:

RUL = RUNTYPEI(model,currenttime)

Description:

This function makes RUL estimats using a Type I (reliability based) prognostics model.

RUL = RUNTYPEI(model,currenttime) makes prognostic estimates for a component or system that has lasted to time currenttime using a Type I model. currenttime can be a scalar indicating the current time for a single component or system, or it may be a vector indicating the current time for a population of components or systems. The RUL estimate for each individual in the population is made independently. currenttime may be input as a column or row vector; the RUL predictions will be returned in the same format.

perc defaults to .95, may be input in percent (1-100) or decimal (0-1)

RUL is calculated as the 0.5-percentile of the conditional probability $R(t|t > T) = \frac{R(t)}{R(T)}$

Example:

```
load failuretimes
whos
```

```
  Name                Size              Bytes  Class     Attributes

  failuretimes      135x1                1080  double
```

```
model = inittypeI(failuretimes);
```

```
currenttime = [0 15 50 100 125];

RUL = runTypeI(model,currenttime)


RUL =


       RUL: [5x1 double]

    StdDev: [5x1 double]

      CI95: [5x2 double]
```

threshGPM

Calculate GPM failure threshold

Syntax:

Threshold = THRESHGPM(prognosticparameters,fit,yt)

Threshold = THRESHGPM(prognosticparameters,fit,yt,thresholdtype,threshcon)

Description:

This function calculates the critical failure threshold for a GPM model based on the full historic failure paths contained in 'progparam'.

Threshold = THRESHGPM(prognosticparameters,fit,yt) determines the critical failure threshold for the paths contained in the cell array 'prognosticparameters'. The function calculates a hard threshold as the conservative 95th percentile of the failure values for the historic paths.

Threshold = THRESHGPM(prognosticparameters,fit,yt,thresholdtype,threshcon) determines the critical failure threshold using the type specified by 'thresholdtype'. 'thresholdtype' may be set to either:

'hard' A constant value is used for the critical failure threshold. This value is the conservative 95^{th} quantile of the failure values for the historic paths.

'pdf' A distribution of critical threshold values is determined. The failure values are assumed to be normally distributed with mean and variance as determined by the historic paths. The threshold distribution is returned as a 1x2 vector of the form [mean standarddeviation].

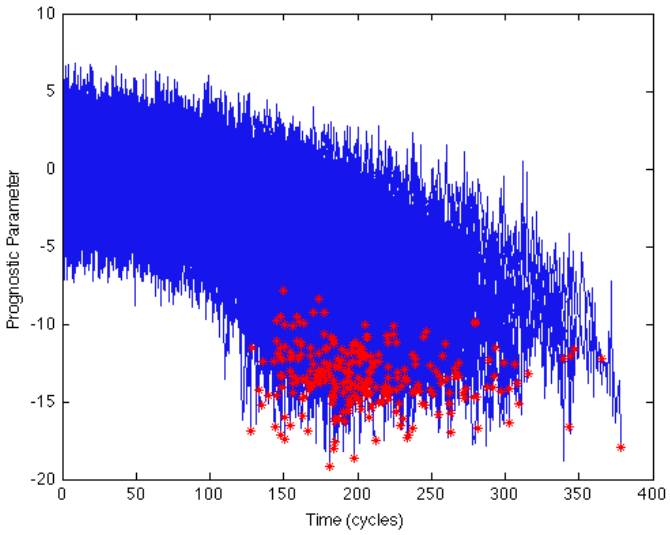
'threshcon' indicates the confidence level at which the threshold is calculated from the failure values.

Example:

```
load PHMchalldata_ga

load PHMchalldata_monitoringresults

whos
```

```
  Name        Size                Bytes  Class     Attributes

  Fhyp        1x260             9060632  cell
  m           1x1                     8  double
  p           1x1                     8  double
  par         1x260              459192  cell
  param       1x1                  1017  struct
  pred        1x260             9060632  cell
  res         1x260             9060632  cell
  t           1x1                     8  double
```

```
prognosticparameter = paramgen(param,res,true);
```




```
[fit ytrans]=fitGPM(prognosticparameter)
```

```
fit =
```

```
    @(x)x.^2    @(x)x    @(x)1

ytrans =

    @(y)y

threshold = threshGPM(prognosticparameter,fit,ytrans)

threshold =

   -9.9524

threshold = threshGPM(prognosticparameter,fit,ytrans,'pdf')

threshold =

  -10.9743    0.5743
```

threshPF

Calculate PF failure threshold

Syntax:

Threshold = THRESHPF(prognosticparameters,fit,yt)

Threshold = THRESHPF(prognosticparameters,fit,yt,thresholdtype,threshcon)

Description:

This function calculates the critical failure threshold for a PF model based on the full historic failure paths contained in 'progparam'.

Threshold = THRESHPF(prognosticparameters,fit,yt) determines the critical failure threshold for the paths contained in the cell array 'prognosticparameters'. The function calculates a hard threshold as the conservative 95th percentile of the failure values for the historic paths.

Threshold = THRESHPF(prognosticparameters,fit,yt,thresholdtype,threshcon) determines the critical failure threshold using the type specified by 'thresholdtype'. 'thresholdtype' may be set to either:

'hard' A constant value is used for the critical failure threshold. This value is the threshcon quantile of the failure values for the historic paths.

'pdf' A distribution of critical threshold values is determined. The failure values are assumed to be normally distributed with mean and variance as determined by the historic paths. The threshold distribution is returned as a 1x2 vector of the form [mean standarddeviation].

'threshcon' indicates the confidence level at which the threshold is calculated from the failure values.

Example:

```
load PFdata

whos
```

```
  Name            Size            Bytes  Class     Attributes

  RUL_actual      1x50              400  double
  gtdata_trn      1x50            65600  cell
  gtdata_tst      1x50            65600  cell
  trn_data        1x50            38088  cell
  tst_data        1x50            30904  cell

[fit xtransform] = fitPF(trn_data);

threshold = threshPF(trn_data,fit,xtransform)

threshold =

    1.3540

threshold = threshPF(trn_data,fit,xtransform,'pdf')

threshold =

    1.5492    0.1044
```

wtd_quantile

Calculate quantiles according to weighted inputs

Syntax:

Y = WTD_QUANTILE(X,W,Q)

Description:

This function estimates quantiles of a sample while considering weighted importance of each sample value

Y = WTD_QUANTILE(X,W,Q) returns the quantiles of the values in X while considering weights of values given in W. Q is a scalar or a vector of cumulative probability values. Both X and W are vectors of the same size. Y is the same size as Q, and Y(i) contains the P(i)-th quantile of X. Linear interpolation is used to estimate the quantile based on the weights in W. If the weights in W are equal, wtd_quatile() returns the same result as quantile()

Example:

```
x = rand(100,1);
weights = rand(100,1);

wtd_quantile(x,weights,0.5)

ans =

    0.5252

wtd_quantile(x,ones(100,1),0.5)
```

```
ans =

    0.5079
```

```
quantile(x,0.5)
```

```
ans =

    0.5079
```

16.4 Appendix 3 References

Abernethy, R. B., *The New Weibull Handbook*, 2nd edn. ISBN 0 9653062 0 8. Abernethy, North Palm Beach, 1996.

Coble, J. and J.W. Hines, "Fusing Data Sources for Optimal Prognostic Parameter Selection." Sixth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies NPIC&HMIT 2009, Knoxville, Tennessee, April 5-9, 2009.

Coble, J.B., "Merging Data Sources to Predict Remaining Useful Life – An Automated Method to Identify Prognostic Parameters." PhD dissertation, University of Tennessee, 2010. <http://trace.tennessee.edu/utk_graddiss/683>.

Cox, D.R. and D. Oakes, Analysis of Survival Data, Chapman and Hall: 1984.

Esary, J.D. and A.W. Marshall, "Shock Models and Wear Processes," *The Annals of Probability* 1 (4) 1973: 627 – 649.

Guess, F. and F. Proschan, "Mean Residual Life: Theory and Applications," FSU Statistics Report M702. AFOSR Technical Report No. 85-178, June 1985.

Gut, A., "Cumulative Shock Models," *Advances in Applied Probability* 22 (2) 1990: 504 – 507.

Hines, J.W. and D. Garvey (2005). "The Development of a Process and Equipment Monitoring (PEM) Toolbox and its Application to Sensor Calibration Monitoring." Fourth International Conference on Quality and Reliability (ICQR4), August 9 - 11, 2005: Beijing, China.

Hines, J.W., J. Garvey, J. Preston, and A. Usynin, "Empirical Methods for Process and Equipment Prognostics," Reliability and Maintainability Symposium RAMS, 2007.

Kumar, D. and B. Klefjo, "Proportional Hazards Model: A Review," *Reliability Engineering and System Safety* 44, 1994: 177 – 188.

Lu, C.J., and W. Meeker, "Using Degradation Measures to Estimate a Time-to-Failure Distribution," *Technometrics* 35 (2) 1993: 161 – 174.

Mallor, F. and J. Santos, "Classification of Shock Models in System Reliability," Monograf´ıas del Semin. Matem. Garc´ıa de Galdeano. 27 2003: 405–412.

Upadhyaya, B.R., M. Naghedolfeizi, and B. Raychaudhuri, “Residual Life Estimation of Plant Components,” *P/PM Technology* June, 1994: 22 – 29.